Homotopy Based Reconstruction from Acoustic Images

Sharma, Ojaswa; Anton, François; Christensen, Niels Jørgen

Publication date:
2010

Document Version
Publisher's PDF, also known as Version of record

Homotopy Based Reconstruction from Acoustic Images

Ojaswa Sharma

Kongens Lyngby 2010
IMM-PHD-2010-234

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-PHD: ISSN 0909-3192

Summary

This thesis presents work in the direction of generating smooth surfaces from linear cross sections embedded in $\mathbb{R}^2$ and $\mathbb{R}^3$ using homotopy continuation. The methods developed in this research are generic and can be applied to higher dimensions as well. Two types of problems addressed in this research are reconstruction from an organised set of linear cross sections and reconstruction from an arbitrary set of linear cross sections. The first problem is looked upon in the context of acoustic signals wherein the cross sections show a definite geometric arrangement. A reconstruction in this case can take advantage of the inherent arrangement. The problem of reconstruction from arbitrary cross sections is a generic problem and is also shown to be solved here using the mathematical tool of continuous deformations. As part of a complete processing, segmentation using level set methods is explored for acoustic images and fast GPU (Graphics Processing Unit) based methods are suggested for a streaming computation on large volumes of data.

Validation of results for acoustic images is not straightforward due to unavailability of ground truth. Accuracy figures for the suggested methods are provided using phantom object with known geometry. The results of the methods shown here can be used to gain objective knowledge about the reconstructed features. It is envisioned that due to the generic nature of the algorithms developed in this research, domains other than fisheries research can benefit from the reconstruction algorithms.

Resumé

Denne afhandling præsenterer arbejde i retningen af at generere glatte overflader fra lineære tværsnit indlejret i $\mathbb{R}^2$ og $\mathbb{R}^3$, der benytter homotopifortsættelse. Metoderne udviklet i denne forskning er generiske og kan også anvendes på højere dimensioner. To slags problemer, som er grebet an i denne forskning, er genopbygning fra et fagorganiseret sæt af lineære tværsnit og genopbygning fra vilkårligt sæt af lineære tværsnit. Det første problem bliver kigget på i konteksten af akustiske signaler hvori tværsnittene viser et bestemt geometrisk orden. En genopbygning i dette tilfælde kan udnytte det iboende orden. Problemet af genopbygning fra vilkårlige tværsnit er et generisk problem og det vises at kunne blive løst her brugende det matematiske redskab af uafbrudte deformeringer. Som en del af en fuldstændig bearbejdning, segmentation der bruger level set metoder bliver udforsket for akustiske billeder og hurtig GPU (Grafikker Processing Unit) baserede metoder bliver foreslået for streaming computation på store volumener af data.

Evaluering af resultater for akustiske billeder er ikke ligetil på grund af mangel af verifikationsdata. Nøjagtighedsmålinger for de foreslåede metoder bliver forsynet ved at bruge fantomobjekt med kendt geometri. Resultaterne af metoderne vist her kan bruges for at opnå objektiv viden om de rekonstruerede træk. Det forventes at på grund af den generiske natur af algoritmerne udviklet i denne forskning, kan andre områder end fiskerierforskning gavne fra genopbygningsalgoritmerne.

Preface

This thesis was prepared at the department of Informatics and Mathematical Modelling, the Technical University of Denmark in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

The thesis deals with object reconstruction from underwater acoustic images and presents novel methods of curve and surface reconstruction using homotopy continuation based techniques. The thesis also addresses a related but more difficult problem of object reconstruction from arbitrary linear cross sections. This work was conducted in collaboration with DTU Aqua and Simrad Kongsberg Maritime. The funding for this work was provided by the Technical University of Denmark.

The work herein presents selected parts of the research carried out during the program of study from 2007 until 2010. The thesis consists of an introductory part containing some background information and an overview of contributions followed by a collection of five research papers. Part of this research was conducted at the Computational Visualization Center (CVC), University of Texas at Austin, USA under the guidance of Professor Chandrajit Bajaj. The project was supervised by Associate Professor François Anton, and co-supervised by Associate Professor Niels Jørgen Christensen from DTU Informatics.

Lyngby, July 2010

Ojaswa

Ojaswa Sharma

Publications

Shortlisted here are scientific publications (journal and conference contributions) made during the course of the PhD program. A list of publications not directly related to the project is also included and referenced at appropriate places throughout this thesis.

Journal

- Ojaswa Sharma, Qin Zhang, François Anton, and Chandrajit Bajaj. Fast, Streaming 3D Levelset on the GPU for Smooth Multi-phase Segmentation. *LNCS Transactions on Computational Science*, 2010. (Under review, Chapter 6 [123])
- Ojaswa Sharma, and François Anton. Homotopy based Surface Reconstruction with Application to Acoustic Signals. *The Visual Computer Journal*, 2010. (Accepted, Chapter 7 [117])
- Ojaswa Sharma, François Anton. Homotopy based 2D Topologic Reconstruction from Arbitrary Linear Cross Sections. *The Visual Computer Journal*, 2010. (Submitted, Chapter 8 [118])

Conference

- Ojaswa Sharma, and François Anton. Homotopic Object Reconstruction Using Natural Neighbor Barycentric Coordinates. *Seventh International*

Symposium on Voronoi Diagrams in Science and Engineering (ISVD), 2010. (Chapter 9 [116])

- Ojaswa Sharma, Qin Zhang, François Anton, and Chandrajit Bajaj. Multi-domain, Higher Order Level Set Scheme for 3D Image Segmentation on the GPU. *23rd IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. ([122])

- Ojaswa Sharma, and François Anton. CUDA based Level Set Method for 3D Reconstruction of Fishes from Large Acoustic Data. *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 17 (WSCG)*, 2009, pp. 153-160. (Chapter 5 [115])

Workshop

- Ojaswa Sharma, Qin Zhang, François Anton, and Chandrajit Bajaj. CUDA Accelerated Multi-domain Volumetric Image Segmentation and Using a Higher Order Level Set Method. *International Workshop on Multidimensional and Mobile Data Model*, 21 - 22 October 2009, UTM Johor, Malaysia.

Additional

- Ojaswa Sharma, François Anton, and Darka Mioc. On the Isomorphism between the Medial Axis and a Dual of the Delaunay Graph. *Sixth International Symposium on Voronoi Diagrams in Science and Engineering (ISVD)*, 2009, pp. 89-95. (Conference paper [119])

- Christopher Gold, Darka Mioc, François Anton, Ojaswa Sharma, and Maciej Dakowicz. A methodology for Automated Cartographic Data Input, Drawing and Editing using Kinetic Delaunay/Voronoi diagrams. *Generalized Voronoi Diagram: A Geometry-Based Approach to Computational Intelligence*, 2008, pp. 159-196, Vol. 158, Springer. (Book chapter [51])

- Ojaswa Sharma, Darka Mioc, and François Anton. Polygon Feature Extraction from Satellite Imagery based on Colour Image Segmentation and Medial Axis. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (Technical Commission III)*, 2008, pp. 235-240, Vol. 37. (Conference paper [121])

- Ojaswa Sharma, Darka Mioc, and François Anton. Feature Extraction and Simplification from Colour Images based on Colour Image Segmentation and Skeletonization using the Quad-Edge data structure. *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 15 (WSCG)*, 2007, pp. 225-232. (Conference paper [120])

Acknowledgements

It is a pleasure to thank those who made this thesis possible. I would first like to show my sincere gratitude to my Ph.D. supervisor François Anton who has been a source of inspiration and has always been available to discuss research problems. I also owe him a note of thanks for being helpful with practical matters during the program of study. Thanks also to my co-supervisor Niels Jørgen Christensen for giving valuable advice on planning the Ph.D. research.

My colleagues from the Image Analysis and Computer Graphics group at DTU Informatics have had a great role in providing a stimulating and friendly atmosphere for good research. Special thanks to my earlier office mates Vesselin and Peter Stanley for being always ready for informative talks. It would be an act of injustice to leave out my buddies Marek, Vedrana and Kasia who have been responsible for introducing levity during tight schedules, deadlines and headaches. Marek, our short breaks of Pool will be very much missed.

My special thanks to Chandrajit Bajaj for supervising my external stay at the University of Texas at Austin. I can assure that my stay turned out to be a great academic experience. I would like to mention that my colleagues at the Computational Visualization Center have been very helpful. Qin Zhang and Ajay, I owe you a lot for being forthcoming in research conversations.

My sincere thanks to Bo Lundgren and Bjarne Stage from DTU Aqua. This work would have been incomplete without your efforts to carry out fisheries experiments and being constructive in discussing the scope of the project. My notable thanks to Lars Nonboe Andersen from Simrad, Norway who has been very encouraging and interested in this project. I have learned a lot from our infrequent but valuable meetings.

Last but not the least, I thank my parents and my wife Shikha for being with me always, providing mental support throughout the project. I dedicate this thesis to my parents and my wife who have been a constant source of inspiration to me.

Contents

Figures

Tables

Algorithms

Symbols

Δ	Laplace operator	7
σ_{bs}	Backscattering cross-section	17
σ_{sp}	Spherical scattering cross-section	17
TS	Target Strength	17
s_v	Volume backscattering coefficient	18
S_v	Volume backscattering strength	18
$\times$	Cartesian product	27
$\mapsto$	Maps to	27
$\in$	Set membership	27
$\mathcal{H}$	Homotopy	27
$\simeq$	Equivalence relation	28
$\implies$	Implies	28
$\circ$	Function composition	28
$\mathbb{R}$	Set of real numbers	31
$\equiv$	Congruence relation	37
$\mathbf{J}$	Jacobian	37
div	Divergence operator	50
∇	Gradient operator	50
lim	Limit	58
ϕ	Signed distance function	59
$\lvert\cdot\rvert$	L_2 norm	59
H	Heaviside function	59
δ_0	Dirac-Delta function	59
δ_ϵ	Regularized Dirac-Delta function	60
H_ϵ	Regularized Heaviside function	60
max	Maximum	62
$\bigcup$	Set-theoretic union	74
$<<$	Right bit-shift operator	77

$\wedge$	Logical And operator	77
min	Minimum	82
d_H	Symmetrical Hausdorff distance	89
sup	Supremum	89
inf	Infimum	89
χ	Characteristic function	101
ker	Kernel (null space)	101
$\bigcap$	Set-theoretic intersection	127
$\otimes$	Tensor product	163

Abbreviations

API	Application Programming Interface, 46
CPU	Central Processing Unit, 48
CUDA	Compute Unified Device Architecture, 3
GPU	Graphics Processing Unit, 3
KB	Kilo Byte, 48
MPI	Message Passing Interface, 45
OpenMP	Open Multi-Processing, 46
PDE	Partial Differential Equation, 44
RAID	Redundant Array of Independent Disks, 26
SIMT	Single-Instruction Multiple-Threads, 48
SM	Streaming Multiprocessor, 48
SONAR	SOund Navigation And Ranging, 12
SP	Scalar Processor, 48
TVG	Time Varied Gain, 20

Chapter 1

Introduction

One of the upfront problems in computer vision and computer graphics today is of object reconstruction from parts. Physical scanning devices provide limited amount of information about the underlying object. This thesis presents research in the direction of object reconstruction by applying principles of homotopy continuation. The less addressed problem of reconstruction from linear cross sections is attempted in the context of fisheries research. Fish school reconstruction from acoustic signals is of importance in estimating fish biomass [126]. Emphasis has been given to formulate homotopy based generic algorithms for reconstruction.

Unlike images, acoustic signals suffer from heavy noise. Part of this research is also dedicated to segmentation of useful features from acoustic signals. These mainly comprise of individual fish and fish school. A good segmentation or noise reduction is a prerequisite for the developed methods of reconstruction.

Echo-sounding devices today are capable of acquiring tremendous amount of data. Digital acoustic data could very well go up to Gigabytes in few hours. Increase in computing power necessitates development of new algorithms that can leverage the computing resources effectively to process huge amounts of information. This objective has been in focus while developing the algorithms.

Lastly, this research also addresses a rather new problem of object reconstruction from arbitrary cross sections. Such a problem holds good importance in its own right since the previous problem can be seen as special case. Algorithms based on homotopy deformation are proposed for smooth reconstruction for such class of problems. Throughout this thesis, the terms "homotopy", "homotopy continuation", and "continuous deformation" have been used interchangeably.

1.1 Objectives

The main objective of this research is to develop algorithms based on homotopy continuation to reconstruct surface models of marine life and objects as seen from an echosounder. Formal scope of this research is to

- Develop methods for underwater 3D reconstruction of fish and fish schools utilizing homotopy continuation techniques. This constitutes construction of a 3D oceanographic view of the water column and the sea floor by reconstituting the space in between the 2D sections with focus on preserving topology of all the contents of the water column.
- Develop effective means of extracting useful features from acoustic signals by removing noise arising from various sources.
- Address a more general problem of object reconstruction from cross sections where no particular ordering within cross sections exist.

Imperative to showcase the adequacy of developed methods is to develop a set of libraries of algorithms and a graphical user interface for reconstruction of the mapped water column and its contents from 2D sectional data.

1.2 Synopsis

This thesis is organized into introductory chapters giving background about the problem domain and the main concepts upon which this research is built followed by chapters based on the scientific publications detailing the main research and its results. Following is an overview of the various chapters.

Chapter 1 introduces this thesis to the reader. A complete thesis flow along with the motives of this research work are given.

Chapter 2 explores the basics of fisheries acoustics. From physics of sound to detecting fishes underwater, this chapter provides an overview of the current state of the fisheries science.

Chapter 3 provides an introduction to the field of homotopy continuation. Homotopy is considered as an effective means of tracing path (or solution) from one system of equations to another in a continuous fashion. Applicability of homotopy continuation and its computational aspects are also explored.

Chapter 4 discusses the problem of feature extraction, grouping and segmentation in general. This problem is discussed in the context of digital images that also include acoustic images. Applicability of the concepts learned so far to the reconstruction problem at hand is investigated. This chapter also serves as a background for the next chapters.

Chapter 5 presents a level set method based approach to segment fishes from a stack of 2D scans. A high performance, streaming level set solver based on the CUDA technology is introduced.

Chapter 6 details a multi-phase, GPU volumetric segmentation scheme that is built on top of the streaming solver for segmenting large volumetric images to generate smooth surfaces.

Chapter 7 explains, in detail, application of homotopy based reconstruction algorithm. Developed homotopies are discussed and necessary mathematical derivations are provided. This chapter concludes with results of reconstruction and an evaluation of the designed algorithms on phantom objects.

Chapter 8 extends the problem of reconstruction to a more general problem of reconstruction from arbitrary cross sections. The complete methodology is described along with its application to object reconstruction in 2D and some accuracy analysis.

Chapter 9 deviates slightly from the algorithm developed in the previous chapter and provides an alternative means of reconstruction utilizing the Voronoi diagram, but still based on homotopy continuation. There is a considerable overlap between this chapter and the previous one, which is primarily due to both being complete and independent publication papers.

Chapter 10 puts the thesis results in a global perspective of this project and concludes the thesis.

The thesis also includes appendices A and B that discuss vital results of the research presented in Chapters 8 and 9 respectively. Appendix C presents the library of algorithms developed as part of this research.

Chapter 2

Fisheries acoustics

Most of the terrestrial monitoring and communication systems make use of electromagnetic waves. Not only these waves travel at the speed of light (3×10^8 ms^{-1}), but also carry information efficiently in the atmosphere and in the space. Unfortunately, the marine realm does not allow for this mode of communication. Propagation of electromagnetic waves in water is accompanied by large energy loss causing fast attenuation that makes them useless for any purpose.

Acoustic waves, on the other hand, can travel very easily in aqueous medium. While electromagnetic waves can hardly penetrate more than a few meters in water, acoustic waves can currently be observed at ranges of up to thousands of kilometers [86]. A sound wave propagating underwater consists of alternating compressions and rarefactions of water particles.

Use of acoustic waves for measuring distant ships was first proposed by Leonardo da Vinci [58]. The speed of sound in water was first measured to be about 1450 ms^{-1} by Colladon and Sturm in 1827 [126]. Practical applications of underwater acoustics were built rather recently during World War I. These included steerable submerged earphones to detect noise sources and locate their direction. Underwater acoustics took a turn around 1915 with the work by a French physicist Paul Langevin who demonstrated the use of transmitted sound signals to actively detect submarines by using a *piezoelectric transducer*[71]. Subsequent applications used this concept and active sonar techniques improved with the advent of better electronic technology and computers. *Acoustic sounders* found widespread use in detecting fish schools in early 1920s. Marine geology benefited from *sidescan sonar* systems to obtain "acoustic images" from the seabed in 1960s. Later in 1970s, seabed acoustic mapping increased dramatically with the emergence of *multibeam echosounders* allowing simultaneous soundings in different directions.

2.1 Underwater acoustics

A sound wave is transmitted in cyclic compressions and rarefactions. Local changes in the pressure are passed on from one point to the surrounding points because of elasticity of the medium. Figure 2.1 schematically shows the components of a sound wave.

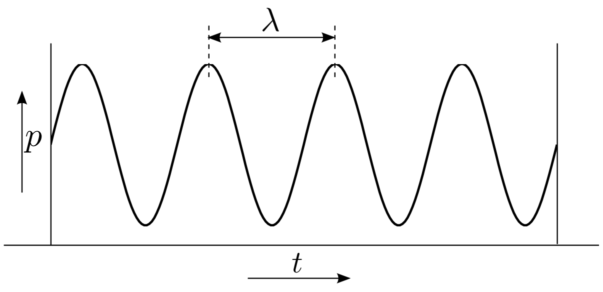


Figure 2.1: *One dimensional acoustic wave.*

Here, λ is the wavelength of the acoustic wave, and p is the pressure.

2.1.1 Wave equation

In one-dimension, an acoustic pressure wave has the following form of partial differential equation [86]

$$\frac{\partial^2 p}{\partial x^2} - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} = 0, \tag{2.1}$$

where p is the acoustic pressure and c is the local propagation velocity of the wave. For a sinusoidal wave of frequency f_0, it can be shown that the solution of (2.1) is

$$p(t) = p_0 \exp\left(2j\pi f_0 \left(t - \frac{x}{c}\right)\right), \tag{2.2}$$

with constant amplitude p_0. This type of wave is referred to as a *plane wave*. For such a wave, the *wave-fronts*, which are surfaces joining continuous loci of peak pressures, are planes. In three dimensions, the wave equation takes the form

$$\Delta p - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} = 0, \tag{2.3}$$

where Δ is the Laplace operator. In an isotropic medium, (2.3) results in a *spherical wave*[86]

$$p(t) = \frac{p_0}{R} \exp\left(2j\pi f_0 \left(t - \frac{R}{c}\right)\right), \tag{2.4}$$

where R is the distance to the source. The wave-fronts are now spheres centered at the source, and the wave amplitude decreases as $1/R$ from its value p_0 considered one meter away from the source.

2.1.2 Intensity and power

A sound wave is associated with an *acoustic energy*. The flux $\mathbf{J}$ is the energy of the wave passing through a unit area orthogonal to the wave-front. The intensity $\mathbf{I}$ is the energy flux per unit time. The total energy $\mathbf{E}$ carried by the wave is the integral of $\mathbf{J}$ with respect to the area over the surface of the wave-front. For a plane wave of amplitude p_0 [86] ,

$$\mathbf{I} = \frac{p_0^2}{2\rho c} \ [\mathrm{W{\cdot}m^{-2}}], \tag{2.5}$$

where ρ is density of the medium. The acoustic power $\mathbf{P}$ received by a surface of area S for a plane wave is [86]

$$\mathbf{P} = \frac{p_0^2 S}{2\rho c} \ [\mathrm{W}]. \tag{2.6}$$

2.1.3 The decibel notation

Since acoustic measurements of pressure or energy vary by large factors, it is convenient to use a logarithmic scale to quantify them, and is noted in *decibels* (dB) units. The decibel corresponds to ten times the base-10 logarithm of ratio of two intensities. Thus for intensities I_1 and I_2, the decibel measurement is [86]

$$r_{dB} = 10 \log_{10} \left(\frac{I_2}{I_1} \right) [dB]. \tag{2.7}$$

Here, I_1 is called the reference level of intensity against which I_2 is observed. In underwater acoustics, the pressure reference is usually one microPascal (1 μPa)[1]. Therefore, pressure measurements are noted as "p dB relative to 1μPa" or "p dB//1 μPa".

2.1.4 Propagation losses

As sound wave propagates, it loses some of its acoustic energy. This happens because the transfer of pressure differences between molecules of water is not 100% efficient - some energy is lost as heat. The energy lost by propagating waves is called *attenuation*. As a sound wave is attenuated, its amplitude is reduced. The level of attenuation of a sound wave is dependent on its frequency - high frequency sound is attenuated rapidly, while extremely low frequency sound can travel virtually unimpeded.

Geometrical spreading losses

An acoustic wave emanating from a source will spread the transmitted acoustic energy on a larger and larger surface. Since, total energy is conserved, the intensity decreases proportionally to the inverse of the surface. This process corresponds to the *geometric spreading loss.* At ranges much higher than the transducer size, also known as the 'far field', the intensity varies with the range R according to the *inverse square law*, otherwise known as *spherical spreading* [126]

$$I = \frac{I_0}{R^2} \tag{2.8}$$

Attenuation losses

Sea water is a dissipative medium and absorbs, via viscosity or chemical reactions, part of the energy of the transmitted wave. The acoustic pressure then

[1]Pascal (Pa) is the SI-derived unit for pressure defined as 1 N·m^{-2}.

decreases exponentially with the distance adding to the spreading loss [86]

$$p(R.t) = p_0 \exp(-\gamma R) \frac{\exp\left(2j\pi f_0 \left(t - \frac{R}{c}\right)\right)}{R}, \tag{2.9}$$

where the attenuation is quantified by the parameter $\gamma($ [Neper$\cdot$m^{-2}]. Attenuation is the most limiting factor in acoustic propagation in the ocean. The amount of attenuation depends on the medium and frequency.

While acoustic energy travels in water, it gets interrupted by sudden change in medium, such as rock or sand. When a moving sound pulse encounters such a medium, some fraction of its energy is propagated into the new material. The energy that is not transmitted into the new medium must go back to the original medium. Some amount of it is reflected off the surface of the material (i.e, bounces off in a certain direction), and the remainder is scattered in all directions (see Figure 2.2). The energy returned to the water is called an *echo*. The echo maintains frequency characteristics of the source wave. The fraction of energy transmitted to the new medium depends on many factors including

- the *impedance* of the new medium (a product of the density of the medium and the speed of sound within it),
- the angle of incidence θ_i of the impinging pulse, and
- the surface roughness of the new medium.

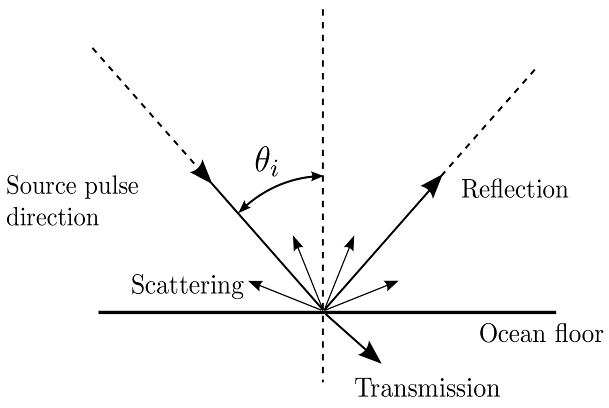


Figure 2.2: *Generation of echo.*

2.2 Instruments

A *sonar* is a device for remotely detecting and locating objects in water using sound. There are two basic types of sonar [126]:

- *Passive sonars* are "listening devices" that record sound emitted by objects in water. Such instruments can be used to detect seismic events, ships, submarines, and marine creatures.
- *Active sonars* are devices that produce sound waves of specific, and controlled frequency and listen for the echoes of these emitted sounds returned from the insonified volume. Sonars measuring ocean depths are active sonars.

To produce sound in water, an echo sounder uses a device called a *transmitter*. Bathymetric sonars require transmitters that can repeatedly produce acoustic pulses with precise, controllable, and repeatable characteristics. Such transmitters are constructed from piezo-electric ceramic, a material that changes its size minutely when a voltage is applied to it. To detect sound in water, *hydrophones* or receivers are used. Hydrophones are analogous to microphones in that they convert sound into electrical signals. Because of the similarity in function, the transmitter and the receiver are often the same pieces of hardware, referred to as the *transducer*.

A pulsed sonar or an *echo sounder* transmits a short burst of sound, called a *pulse* or a *ping*, consisting of several cycles at the sonar operating frequency. The transmitted pulse travels away from the transducer. Once it encounters a target, some energy is reflected back as echo, which travels back to the transducer. If the echo is detected at time t_e after the transmission, the range R is given by [126]

$$R = \frac{ct_e}{2}. \tag{2.10}$$

In order to resolve two objects at ranges R_1 and R_2, the difference $(R_2 - R_1)$ must be large enough for these two echoes to not overlap. Thus,

$$R_2 - R_1 > \frac{c\tau}{2}, \tag{2.11}$$

where τ is the pulse duration [126]. Basic operation of an echosounder is shown in Figure 2.3.

Two main types of active *multi-beam sonars* of relevance to this research are the *sector scanner* SeaBat 7128 from RESON and the *three-dimensional sonar* MS70 from Simrad.

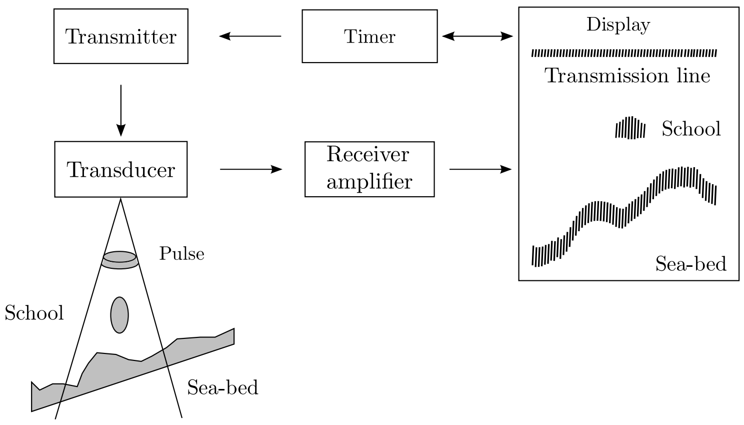


Figure 2.3: *Concept of echosounding (adapted from Simmonds and MacLennan [126, chap. 3]).*

2.2.1 Sector scanners

In a searchlight or side-scan sonar, the transducer moves or rotates to insonify and scan different volumes of water. A sector scanner does this rapidly and without moving the instrument. The transducer consists of an array of multiple elements arranged in a line. The transmitted pulse is applied to one central element generating a wide beam upon transmission. The reflected signal is received as thin scanning beams. The scanning beam is electronically steered so that it sweeps a sector in much less time compared to the pulse duration. This so called process of *beamforming* is explained in section 2.3.

Over the past years various sector scanning multi-beam sonars have been successfully used in fisheries. SeaBat 7128 from RESON is a high-resolution, forward-looking sonar system that operates at 200 kHz or 400 kHz frequency. Depth ratings available with this sonar are up to 6000 meters [110]. Table 2.1 lists some of the specifications of this sonar. The transducer is shown in Figure 2.4.

2.2.2 Three-dimensional sonar

The MS70 sonar from Simrad provides a three-dimensional observation from each ping. It uses a technique called “Frequency Rotated Sector Transmission”

Table 2.1: *SeaBat 7128 multi-beam sonar [110].*

Specification	Value
Operating frequency	200 or 400 kHz
Number of beams	256
Beam Width	0.5°
Range	200 m at 400 kHz, 500 m at 200 kHz
Range resolution	2.5 cm

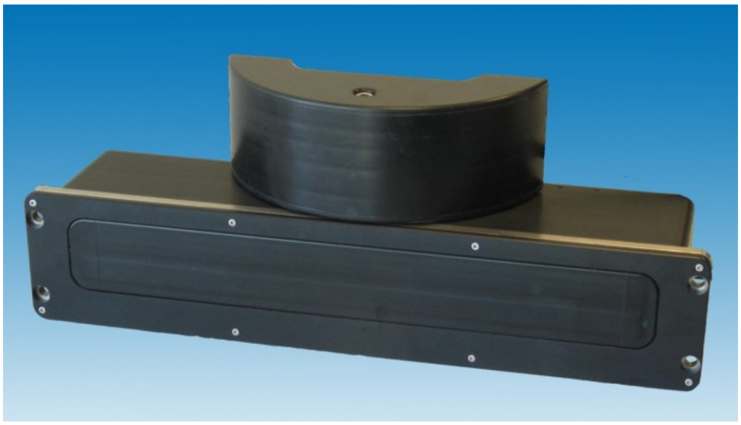


Figure 2.4: *SeaBat 7128 multi-beam sonar transducer[110].*

[100]. The operational range of frequencies is 70 to 120 KHz divided into 20 sub-bands. The system transmits at the central frequency of each sub-band insonifying 20 horizontal sectors of 60° width. On the receiving side, it forms 25 narrow beams for each sector, thus giving a total of 500 narrow receiving beams. Table 2.2 lists some of the specifications of this sonar. The transducer is shown in Figure 2.5.

2.3 Beamforming

An isotropic acoustic source is not ideal for a depth-sounding sonar for two reasons

Table 2.2: *MS70 3D multi-beam sonar [128].*

Specification	Value
Operating frequency	75 to 112 kHz
Number of beams	500, organized as a matrix
	Horizontal: 25, vertical: 20
Beam Opening angle	Horizontal: 3°, vertical: 4°
Operating sectors	Horizontal: 60°, vertical: 45°
Number of transducer elements	800

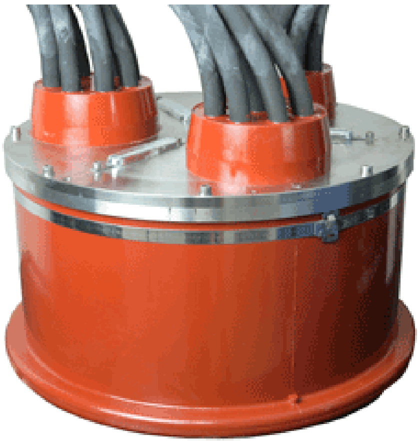


Figure 2.5: *MS70 3D multi-beam sonar transducer[128].*

- *No directivity*: a spherical wavefront strikes the ocean floor in all directions. There is no way to determine the direction of the return echoes.
- *Isotropic power distribution*: the power of the transmitted pulse is distributed equally in all directions thus leaving very little power for the features of interest.

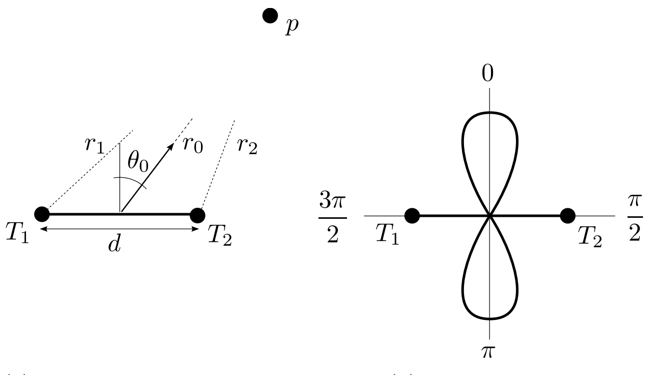
Groups of isotropic sources can be used to transmit non-isotropic waves by means of an interference pattern of spherical waves. Directed pulses can be used to insonify specific areas on the sea floor with higher energy. Transmission directivity allows concentration of the transmitted energy into a particular angular sector, thus increasing local acoustic pressure.

Consider two transmitting transducers T_1 and T_2 placed d distance apart. For a far-off point p ($r_0 \gg d$, see Figure 2.6(a)), distant r_1 and r_2 from T_1 and T_2 respectively, the path difference $|r_2 - r_1|$ determines the type of interference at it. For a constructive interference at p [86],

$$r_2 - r_1 = d\sin(\theta_0) = n\lambda,\ n \in \mathbb{Z}^+, \tag{2.12}$$

while for a destructive interference at p,

$$r_2 - r_1 = d\sin(\theta_0) = \left(n + \frac{1}{2}\right)\lambda,\ n \in \mathbb{Z}^+. \tag{2.13}$$



(a) Geometry for interference (b) Beam pattern

Figure 2.6: *Acoustic interference by two transducers.*

For a typical sonar, the spacing d is half the wavelength, $\lambda/2$. Therefore, constructive interference occur at angles 0 and π, while destructive interference occur at angles $\pi/2$ and $3\pi/2$ (i.e., at the axis of separation). A polar plot of the resulting amplitude with the center of transducers as origin is what is called a *beam pattern*. Figure 2.6(b) shows the beam pattern resulting from a two transducer geometry. The plot clearly shows that the bulk of the energy is transmitted in a direction orthogonal to the axis of separation of the transducers. The beam pattern provides a measure of the directivity.

Real transducer arrays generally have more than two transducer elements and have complicated beam patterns. Figure 2.7 shows beam pattern of a multiple element line array. The bulk of the energy is in what is called the *main lobe*. The direction of the peak energy projection is called the *maximum response axis*. The *width* or *beam solid angle* of the main lobe is twice the angle from the axis to the half power point on the pattern.

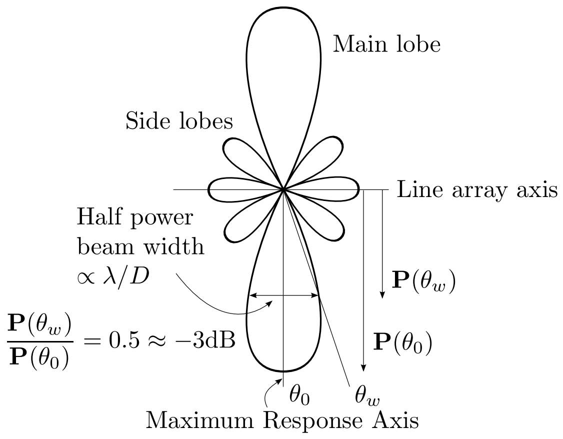


Figure 2.7: *Beam pattern of a multiple-element line array [63].*

Side lobes in a beam pattern are unavoidable, although the energy distributed in them can be lowered to a certain extent by projecting stronger signals from the individual elements in the center of the array. This technique of reducing side lobe is called *shading*. One popular shading scheme is called the *Dolph-Chebyshev* shading [71].

A hydrophone array can be used to listen to the acoustic echoes, just like a transmitter is used to generate pulses. The sensitivity of a hydrophone array follows the beam pattern of a corresponding transmitter array, by virtue of the principle of reciprocity. Therefore, a hydrophone array is very sensitive to echoes in the direction of the maximum response axis. For an array made of independent discrete hydrophones, appropriate phase or time shifts can be imposed on the hydrophones to steer the main lobe of the array in the direction of choice [63]. This process is called *beamforming*, and is used both on transmission and on reception.

In three-dimensions, the beam pattern shown in Figure 2.7 is rotationally symmetric about the axis of separation. Therefore, an emitter array insonifies a strip at the bottom and consequently the hydrophone array cannot distinguish the location of a particular echo along the strip (see Figure 2.8). If, however, the emitter and the hydrophone arrays are arranged orthogonal to each other, the hydrophone array can observe a small area at the bottom of the sea that is formed by the intersection of the two strips. This perpendicular arrangement of the two arrays is called a *Mills Cross* (see Figure 2.9) [63]. Furthermore, since the hydrophone array can be steered by introducing a time delay, multiple steered beams can be used to receive echoes from discreet locations all along the insonified strip (see Figure 2.10).

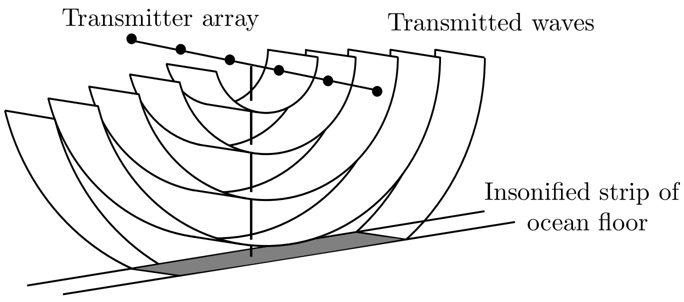


Figure 2.8: *Insonified strip by an array of transducers [63].*

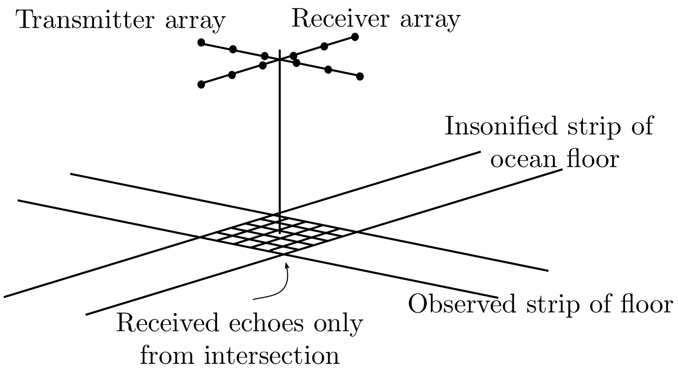

Transmitter array
Receiver array
Insonified strip of ocean floor
Observed strip of floor
Received echoes only from intersection

Figure 2.9: *Mills Cross arrangement [63].*

2.4 Fish measurement

Marine fisheries make extensive use of underwater acoustic techniques. Modern fishing and oceanographic vessels are generally equipped with many high end sonar systems. Apart from estimating the fish abundance and locating fish schools to capture fish, other applications include identifying fish species and evaluating biomass.

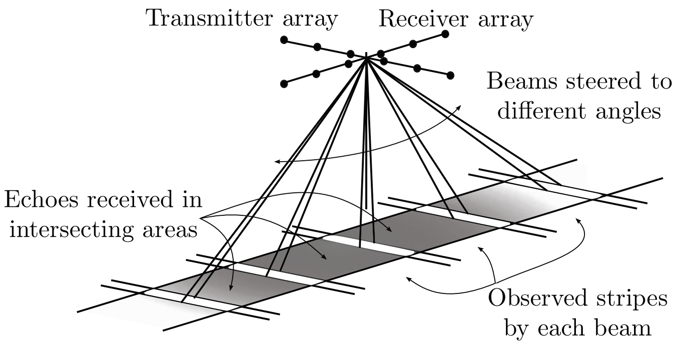

Transmitter array
Receiver array
Beams steered to different angles
Echoes received in intersecting areas
Observed stripes by each beam

Figure 2.10: *Mills Cross with multiple steered beams [63].*

2.4.1 Target strength

Target strength is a logarithmic measure of the proportion of the incident energy which is backscattered by the target. A related quantity is the *backscattering cross-section*, σ_{bs}, measured in units of area. It is defined in terms of the intensities of the incident and the backscattered waves. Assuming I_i is the incident acoustic intensity on a target, I_{bs} is the intensity at the midpoint of the backscattered pulse, and R is the distance from the target (much greater than the linear size of the target), then σ_{bs} is defined as [126]

$$\sigma_{bs} = \frac{R^2 I_{bs}}{I_i}. \tag{2.14}$$

Another measure of same quantity is the *spherical scattering cross-section* [126],

$$\sigma_{sp} = 4\pi\sigma_{bs}. \tag{2.15}$$

σ_{sp} is an older concept that comes from the idea that the scattered intensity is isotropic. The target strength is then defined as the backscattering cross-section expressed in decibels [126]

$$TS = 10\log_{10}(\sigma_{bs}) = 10\log_{10}(\sigma_{sp}/4\pi), \tag{2.16}$$

with σ_{bs} or σ_{sp} specified in m^2. The logarithmic TS is convenient in acoustics since acoustic backscattering strengths vary a lot between aquatic organisms.

2.4.2 Volume scattering coefficients

For a school of fish or organisms detected in high density, the echoes combine to form a return signal that is continuous with varying amplitude. Individual echoes cannot be discerned but the echo intensity is still a measure of the biomass in the water column. A basic acoustic measurement in such a case is the *volume backscattering coefficient* [126], s_v defined as

$$s_v = \frac{\sum \sigma_{bs}}{V_0}, \tag{2.17}$$

where the sum is taken over all the discrete targets contributing to the echoes from the sampled volume V_0. The equivalent logarithmic measure is the *volume backscattering strength* S_v[126],

$$S_v = 10 \log_{10}(s_v), \ [\text{dB re 1 m}^{-1}]. \tag{2.18}$$

2.4.3 Range compensation

Effects like beam spreading and absorption decrease the amplitude of the echo with the range as sound propagates in water. Modern equipments include a *time-varied-gain* (TVG) amplifier to account for the propagation losses. The gain is increased in proportion to the TVG function $a(t)$, where t is the time from the start of the transmitter pulse (see Figure 2.11). For an isolated target whose range is much greater than the pulse length, a suitable TVG function is [126]

$$a(t) = (ct)^2 \exp\left(\frac{\beta ct}{2}\right), \tag{2.19}$$

where β is the absorption coefficient. This function is commonly known as '40 log R' TVG. This is obtained by substituting $R = ct/2$, and calculating the TVG expressed in decibels as $20 \log_{10}(a(t))$.

In case of multiple targets distributed randomly over the beam, a different TVG function is more suitable. This function, known as '20 log R', is

$$a(t) = (ct) \exp\left(\frac{\beta ct}{2}\right). \tag{2.20}$$

It takes into account the fact that a larger number of targets compensates the transmission losses to some extent.

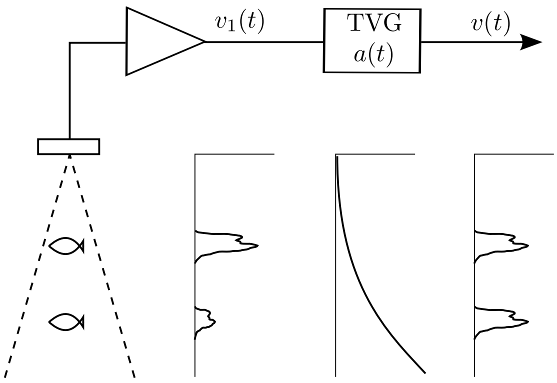


Figure 2.11: *Range compensation by TVG for single targets [126]. $v_1(t)$ is the uncompensated signal and $v(t) = a(t)v_1(t)$ is the range compensated signal for similar targets.*

2.4.4 Reverberation and noise

Reverberation refers to the echoes from unwanted targets in the insonified volume. It consists of contributions from parasitic echoes from the propagating medium overlaid on the useful signal. The characteristics and origins of reverberation are the same as that of the useful signal that makes its removal difficult.

The causes of underwater noise can be grouped into following categories

- *Ambient noise* originates from outside the system either from natural or man-made causes. It is independent of the sonar system itself. The physical origins of the ambient noise include very low frequency noise generated by remote seismic or volcanic activity, shipping or other industrial activity in the water, and surface agitation by wind.
- *Self-noise* is the noise caused by the acoustic system, the supporting platform, or the system electronics. The main sources of such noise are the propeller noise, turbulence generated by the flow of water, and machine noise (engine, reduction gears, generators, and hydraulic machinery).
- *Reverberation* As discussed before, such noise is caused by the parasite echoes. These echoes could be laud enough to mask the detection of expected target echoes.

- *Acoustic interference* is generated by other acoustic systems nearby the system. Existence of several transmitting acoustic transducers in a restricted space is most likely to cause interference, where the level transmitted by each is enough to affect the others completely.

2.4.5 Near field and far field

The insonified volume can be divided into two zones depending on the distance of the observation point from the transducer [86]

- *Near field* (or Fresnel zone): The range dependence of intensity in this region is very complicated, and it varies in an oscillatory manner. The contributions from different points on the transmitter face are strongly out of phase with each other. The average intensity in this region decreases more slowly than the spherical spread in $1/R^2$.
- *Far field* (or Fraunhofer zone): The element wave-fronts are nearly parallel in this region. The sound field is more uniform away from the transducer. The intensity of the field decreases monotonously with distance, and the inverse square law applies.

If a is the linear distance across the transducer face, the boundary between the near and far fields is approximately at the range [126]

$$R_b = \frac{a^2}{\lambda} \tag{2.21}$$

2.5 Thresholding

A threshold is generally applied to remove any unwanted signal detected by the echosounder. The unwanted signal could be due to electrical noise in the equipment, acoustic reverberation, or the merged echoes from non-target species. When a threshold is applied, any target echoes smaller than the threshold are also ignored. Therefore, the results of thresholding are biased because some proportion of the target population is rejected [126]. Numerous experiments have been conducted to determine appropriate thresholds to increase the signal to noise ratio, while preserving the desired signal (see, for example, [108, 139, 72]).

Other image segmentation techniques have also been successfully tried to extract useful information from sonar images. These include graph cuts [91, 92], mean-shift clustering [23], and anisotropic diffusion for speckle reduction [73]. In this research, a new approach to feature extraction has been adopted for noise suppression by using level set based segmentation. The closest work to this is by Lianantonakis and Petillot [82] in which a level set based technique is applied for textural segmentation of side-scan imagery of the sea floor. In this research, a level set approach is explored to segment features and GPU based methods are developed for faster processing [115, 123]. This is the topic of discussion in Chapters 5 and 6.

2.6 Datasets

As noted before in section 2.2, this research is mostly concerned experimentally with acoustic measurement of fish and fish schools from the instruments SeaBat 7128 from RESON and MS70 from Simrad. The details of the data acquisition, experimental setup and sample imagery is presented in this section.

2.6.1 SeaBat 7128 dataset

In a series of sonar experiments conducted at the Nordsøen oceanarium by a group of scientists from DTU Aqua, acoustic data was collected using the SeaBat 7128 instrument. Nordsøen oceanarium is a public aquarium located at the shores of the North Sea in Hirtshals, North Jutland, Denmark. It is the largest aquarium in Northern Europe and contains twelve tanks ranging in size from 4,000 to 16,000 liters. The oceanarium, that was the ground for the experiments, contains 4.5 million liters of sea water and about 2000~3000 fish, shoals and two sunfish (see Figure 2.12). It is an eight meter deep tank with an elliptical cross section measuring 22 by 33 meters [98].

The experimental setup consisted of a SeaBat 7128 sonar mounted on a vertical rail installed on a wooden raft as shown in Figure 2.13. The whole system was kept afloat on the surface of the water tank and the acoustic instrument could be moved along the rail with the help of an electric motor. A video camera looking vertically down into the water tank was also installed to monitor part of the underwater volume scanned by the acoustic instrument. The video recording was intended to provide supplemental information about the scanned fish species and fish movement behavior.

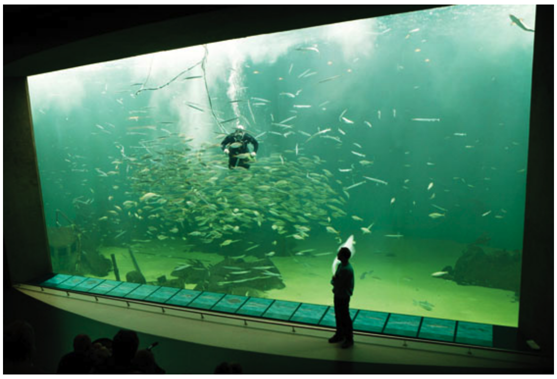


Figure 2.12: *Nordsøen oceanarium [98].*

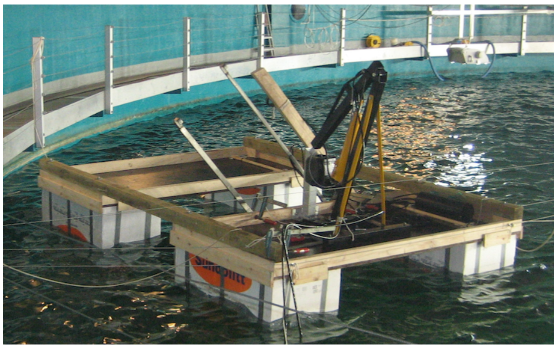


Figure 2.13: *Raft for carrying the transducer.*

A total of eight configurations of the acoustic instrument were explored in different experiments

- *Experiment 1* was to scan the oceanarium keeping the transducer looking vertically downwards with the video camera recording part of the insonified volume (see Figure 2.14(a)).
- *Experiment 2* consisted of the same configuration with the transducer moving along the rail back and forth (see Figure 2.14(b)).
- *Experiment 3* was to scan the water from bottom looking up while keeping the transducer static (see Figure 2.14(c)).

- *Experiment 4* explored sideways scan of the water with instrument positioned as shown in Figure 2.14(d).
- *Experiment 5* consisted a tilted view of the water with transducer rotated at an angle of about 27° from vertical and looking at the bottom of the tank (see Figure 2.14(e)).
- *Experiment 6* was conducted to calibrate the sonar instrument with a standard target of copper sphere of radius 30.0 mm.
- *Experiment 7* included scans with low power and gain settings so as to reduce side lobe effects.
- *Experiment 8* was to scan the bottom of the oceanarium with the transducer kept horizontal and static (see Figure 2.14(f)).
- *Experiment 9* consisted of the same horizontal configuration but with the transducer moving up and down to scan a larger volume of the oceanarium (see Figure 2.14(g)).
- *Experiment 10* consisted of a tilted scan of the water with the rail tilted by small angle from the vertical (see Figure 2.14(h)).

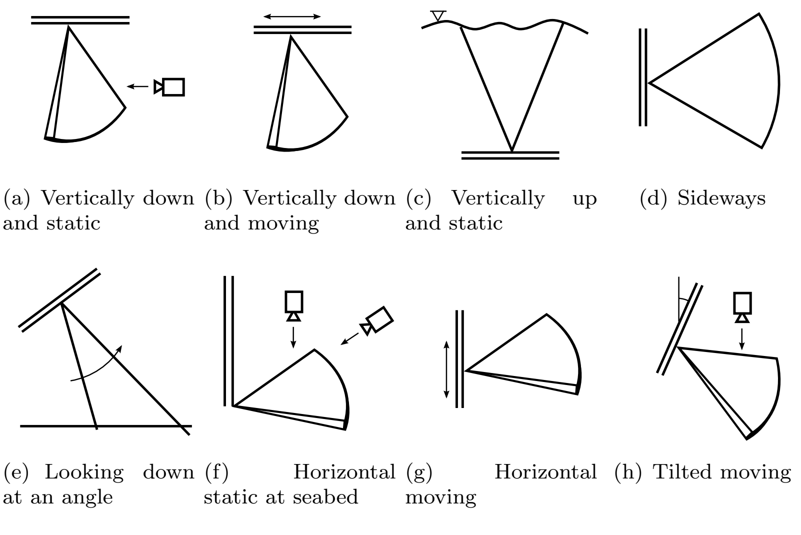

(a) Vertically down and static
(b) Vertically down and moving
(c) Vertically up and static
(d) Sideways
(e) Looking down at an angle
(f) Horizontal static at seabed
(g) Horizontal moving
(h) Tilted moving

Figure 2.14: *Various transducer configurations in the experiment.*

Acoustic data from these experiments is used in this research for reconstruction of the scanned fish. Monitoring and data storage system for these experiments consisted of video screens for live viewing of the captured data and high capacity RAID storage for keeping acoustic signals coming from the sonar instrument (see Figure 2.15). Typical frames from the video camera looking downward from top of the oceanarium and the sonar instrument looking horizontally are shown in Figures 2.16(a) and 2.16(b) respectively. Note that the wall of the elliptical tank can be seen on the right in the sonar image.

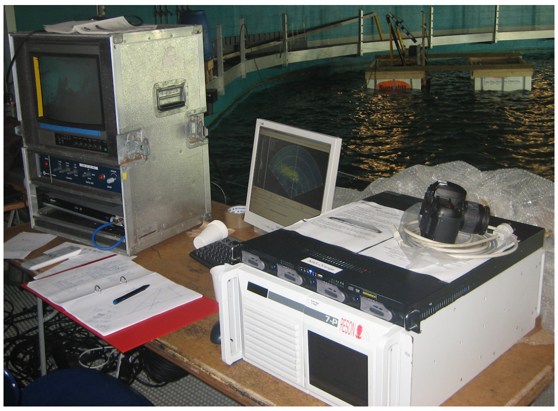


Figure 2.15: *Video and storage system for sonar data acquisition.*

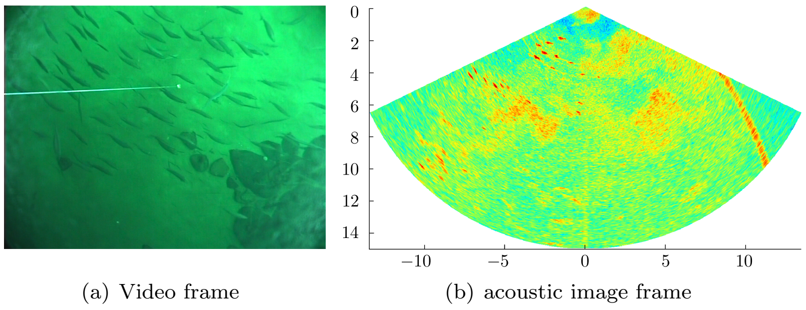


Figure 2.16: *Typical video and acoustic image frames.*

2.6.2 MS70 dataset

The MS70 3D data was received from Simrad as part of collaboration in the current research. One of the pings from the dataset is shown in Figure 2.17 as a 3D volume raycasting. The topmost sectors of the data contain high reflections from the water surface. The data was collected in a fjord where the data was collected and captures a moving school of Sprat.

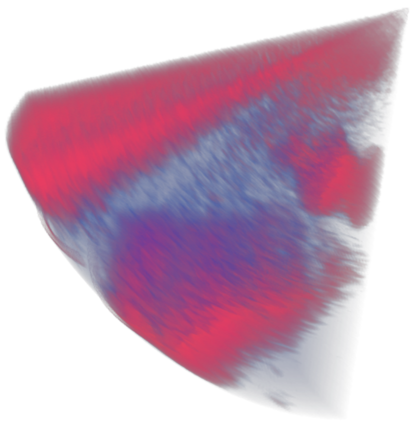


Figure 2.17: *A ping from the 3D sonar.*

Chapter 3

Homotopy continuation

Homotopy is concerned with identification of paths that can be continuously deformed into each other [65]. Such paths are then considered equivalent. This type of equivalence was given in late 1920's. Originally, homotopy was used as a tool to decide weather two paths with same end-points would lead to the same result of integration. The use of homotopies can be tracked back to the works of Poincaré (1881-1886), Klein (1882-1883), and Berstein(1910) [65].

A homotopy is defined as a continuous map between two continuous functions in a topological space. A homotopy can, therefore, be viewed as a *continuous deformation*. The use of deformations to solve non-linear systems of equations may be traced back at least to Lahaye (1934) [3].

Definition 3.1 A homotopy between two continuous functions f_0 and f_1 from a topological space $\mathcal{X}$ to a topological space $\mathcal{Y}$ is defined as a continuous map $\mathcal{H} : \mathcal{X} \times [0,1] \mapsto \mathcal{Y}$ from the product space $\mathcal{X}$ with the unit interval $[0,1]$ to $\mathcal{Y}$ such that

$$\mathcal{H}(\mathbf{x}, 0) = f_0, \text{ and} \tag{3.1}$$

$$\mathcal{H}(\mathbf{x}, 1) = f_1, \tag{3.2}$$

where $\mathbf{x} \in \mathcal{X}$. The second parameter of $\mathcal{H}$, also called the *homotopy parameter*, allows for a continuous deformation of f_0 to f_1. f_0 and f_1 are also known as the *initial map* and the *terminal map* respectively.

Two continuous functions f_0 and f_1 are said to be *homotopic*, denoted by $f_0 \simeq f_1$, if and only if there is a homotopy $\mathcal{H}$ taking f_0 to f_1. Being homotopic is an equivalence relation on the set $\mathrm{C}(\mathcal{X}, \mathcal{Y})$ of all continuous functions from $\mathcal{X}$ to $\mathcal{Y}$.

Theorem 3.2 *Homotopy is an equivalence relation on* $\mathrm{C}(\mathcal{X}, \mathcal{Y})$ *[34].*

Proof. This can be verified by showing that $\simeq$ is reflexive, symmetric, and transitive.

Reflexivity ($f_0 \simeq f_0$). The map $\mathcal{H}(\mathbf{x}, \lambda) = f_0(\mathbf{x}), \mathbf{x} \in \mathcal{X}, \lambda \in [0,1]$ is a homotopy from f_0 to f_0.

Symmetry ($f_0 \simeq f_1 \implies f_1 \simeq f_0$). Let $\mathcal{H} : \mathcal{X} \times [0,1] \mapsto \mathcal{Y}$ be a homotopy from f_0 to f_1. Then the map $\mathcal{G} : \mathcal{X} \times [0,1] \mapsto \mathcal{Y}$,

$$\mathcal{G}(\mathbf{x}, \lambda) = \mathcal{H}(\mathbf{x}, 1 - \lambda)$$

is a homotopy from f_1 to f_0.

Transitivity ($f_0 \simeq f_1$ and $f_1 \simeq f_2 \implies f_0 \simeq f_2$). Let $\mathcal{H} : \mathcal{X} \times [0,1] \mapsto \mathcal{Y}$ be a homotopy from f_0 to f_1 and $\mathcal{G} : \mathcal{X} \times [0,1] \mapsto \mathcal{Y}$ be a homotopy from f_1 to f_2. Then the map $\mathcal{F} : \mathcal{X} \times [0,1] \mapsto \mathcal{Y}$,

$$\mathcal{F} = \begin{cases} \mathcal{H}(\mathbf{x}, 2t) & \text{if } 0 \leq t \leq 1/2, \text{ and} \\ \mathcal{G}(\mathbf{x}, 2t-1) & \text{if } 1/2 \leq t \leq 1 \end{cases}$$

is a homotopy from f_0 to f_2 [34]. □

Definition 3.3 A function $f_0 : \mathcal{X} \mapsto \mathcal{Y}$ is *null-homotopic* if it is homotopic to a constant function c, that is, $f_0 \simeq c$.

If a space $\mathcal{X}$ has the property that $\mathrm{id}_{\mathcal{X}}$, the identity map on $\mathcal{X}$, is null-homotopic, then $\mathcal{X}$ is contractible [96]. A contractible space is one that can be continuously shrunk to a point.

Proposition 3.4 *Let $f_0, f_1 : \mathcal{X} \mapsto \mathcal{Y}$ and $g_0, g_1 : \mathcal{Y} \mapsto \mathcal{Z}$ be continuous functions, and let $g_0 \circ f_0, g_1 \circ f_1 : \mathcal{X} \mapsto \mathcal{Z}$ be the respective composite maps. If $f_0 \simeq f_1$ and $g_0 \simeq g_1$, then $g_0 \circ f_0 \simeq g_1 \circ f_1$ [34].*

Proof. Let $\mathcal{H} : \mathcal{X} \times [0,1] \mapsto \mathcal{Y}$ be a homotopy between f_0 and f_1 and $\mathcal{G} : \mathcal{Y} \times [0,1] \mapsto \mathcal{Z}$ be a homotopy between g_0 and g_1. Define a map $\mathcal{F} : \mathcal{X} \times [0,1] \mapsto \mathcal{Z}$ by

$$\mathcal{F}(\mathbf{x}, \lambda) = \mathcal{G}(\mathcal{H}(\mathbf{x}, t), t),$$

for $\mathbf{x} \in \mathcal{X}$. It can be seen that

- $\mathcal{F}$ is continuous,
- $\mathcal{F}(\mathbf{x}, 0) = \mathcal{G}(\mathcal{H}(\mathbf{x}, 0), 0) = \mathcal{G}(f_0(\mathbf{x}), 0) = g_0(f_0(\mathbf{x}))$, and
- $\mathcal{F}(\mathbf{x}, 1) = \mathcal{G}(\mathcal{H}(\mathbf{x}, 1), 1) = \mathcal{G}(f_1(\mathbf{x}), 1) = g_1(f_1(\mathbf{x}))$.

Therefore, $\mathcal{F}$ is a homotopy between $g_0 \circ f_0$ and $g_1 \circ f_1$. □

Definition 3.5 For two homeomorphisms $h_0, h_1 : \mathcal{X} \mapsto \mathcal{Y}$, an *isotopy* is a homotopy, $\mathcal{H}(\mathbf{x}, \lambda), \mathbf{x} \in \mathcal{X}, \lambda \in [0,1]$, such that for each fixed λ, $\mathcal{H}(\mathbf{x}, \lambda)$ gives a homeomorphism.

Requiring that two homeomorphisms be isotopic is a stronger requirement than that they be homotopic.

In this thesis, the focus is on defining homotopies for smooth reconstruction between cross sectional information. Therefore, numerical/computational aspects of homotopy continuation based methods are considered in next section.

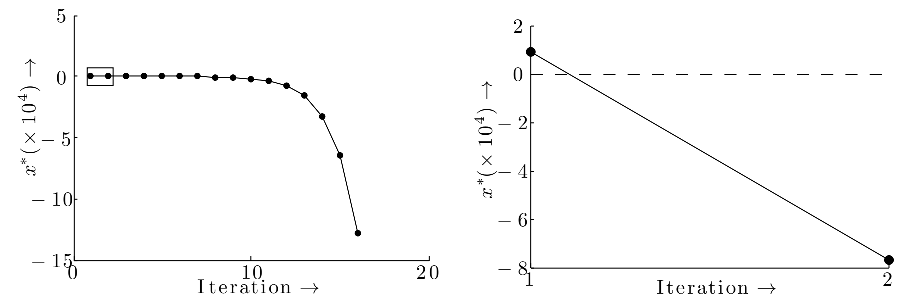
3.1 Motivation

The strength of homotopy continuation can be seen in solving non-linear system of equations. The main difficulty with conventional numerical non-linear solvers is in choosing a good initial approximation of the solution. For many such algorithms, convergence is guaranteed for a restricted set of initial vectors. Therefore, in numerical solutions, homotopy methods provide global convergence (or a greatly expanded domain of convergence), as opposed to local convergence of most iterative methods. The basic idea behind using continuation in numerical computation is to solve a series of problems as the homotopy parameter is slowly varied, using a locally convergent iterative technique for each problem, and the solution to the previous problem as starting point for the current problem. The very first problem in the series is chosen to be a simple problem for which the solution is known [3, 76, 38, 140].

The strength of a homotopy method can be illustrated with a small example. Consider the solution to the equation $f(x) = 0$, where

$$f(x) = \frac{x}{1+x^2}.$$

Clearly, $x = 0$ is the only solution to this equation. However, beginning with an initial guess of $x_0 = 0.5$, Newton's method yields a diverging sequence (see Figure 3.1). Here the problem comes from the fact that f approaches 0 asymptotically as x goes to $-\infty$. Newton's method skips the solution after the first iterate (see Figure 3.1(b)) and then onward seeks the solution in the wrong direction.



(a) Newton iterations

(b) Closer look at the solution after first and second iterates.

Figure 3.1: *Diverging solution with conventional Newton's method.*

The first derivative of f is

$$f'(x) = \frac{1-x^2}{(1+x^2)^2}.$$

$f(x)$ has stationary points at $x = \pm 1$. These stationary points cannot serve as initial guess since they will require a division by zero in the iterate. To get rid of this difficult situation, a homotopy based solution can be sought. Beginning again with our initial guess $x_0 = 0.5$, we formulate a homotopy

$$\begin{aligned}\mathcal{H}(x,\lambda) &= \lambda f(x) + (1-\lambda)(f(x) - f(x_0)) \\ &= f(x) + (\lambda - 1) f(x_0),\end{aligned}$$

starting with the initial map $(f(x)-f(x_0))$ that has a trivial zero. The homotopy parameter λ is discretized as $\{\lambda_k\}, k \in [0, n-1]$ such that $\lambda_k < \lambda_{k+1}$, $\lambda_0 = 0$, and $\lambda_{n-1} = 1$. The homotopy is then solved for every λ_k such that the solution x_k^* of $\mathcal{H}(x, \lambda_k) = 0$ serves as the initial guess x_0 for $\mathcal{H}(x, \lambda_{k+1}) = 0$. In this way, a path of solutions $\{x_k^*\}$ is obtained that connects our initial guess $x_0 = 0.5$ to the solution $x = 0$ of $f(x) = 0$ obtained with $\lambda = 1$. Figure 3.2 shows the resulting path.

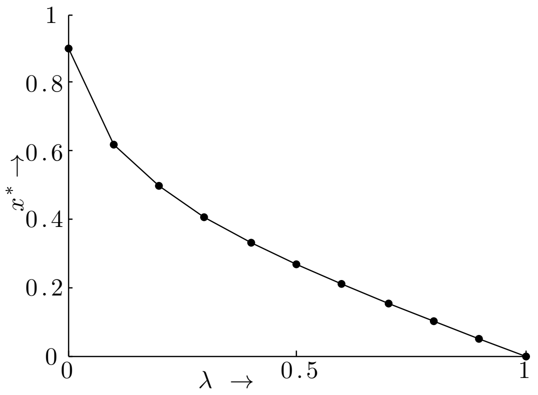


Figure 3.2: *Homotopy path.*

3.2 Numerical continuation method

For a given non-linear system of n equations

$$f(\mathbf{x}) = 0 \tag{3.3}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a smooth mapping, as seen previously, the solution via a Newton-type method requires a good choice of initial guess $\mathbf{x}_0$. In absence of such a choice, a homotopy $\mathcal{H}(\mathbf{x}, \lambda) : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}^n$ can be formulated such that

$$\begin{aligned} \mathcal{H}(\mathbf{x}, 1) &= f(\mathbf{x}), \\ \mathcal{H}(\mathbf{x}, 0) &= g(\mathbf{x}), \end{aligned} \tag{3.4}$$

where $g : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a (trivial) smooth map with known zeros. A few choices of homotopies are

- A *convex homotopy* such as

$$\mathcal{H}(\mathbf{x},\lambda) = \lambda f(\mathbf{x}) + (1-\lambda)g(\mathbf{x}), \lambda \in [0,1]. \tag{3.5}$$

- A *global homotopy* such as

$$\mathcal{H}(\mathbf{x},\lambda) = f(\mathbf{x}) - (1-\lambda)f(\mathbf{x}_0), \lambda \in [0,1], \tag{3.6}$$

  for some $\mathbf{x}_0$ as an initial guess (the choice of which is less restrictive than in a Newton-type method).

If D$\mathcal{H}$, the total Frechet derivative of $\mathcal{H}$ with respect to its variables (i.e, the $n \times n+1$ Jacobian matrix) has full rank n, then according to the implicit function theorem, a curve $c \in \mathcal{H}^{-1}(0)$ will be a smooth curve without intersections.

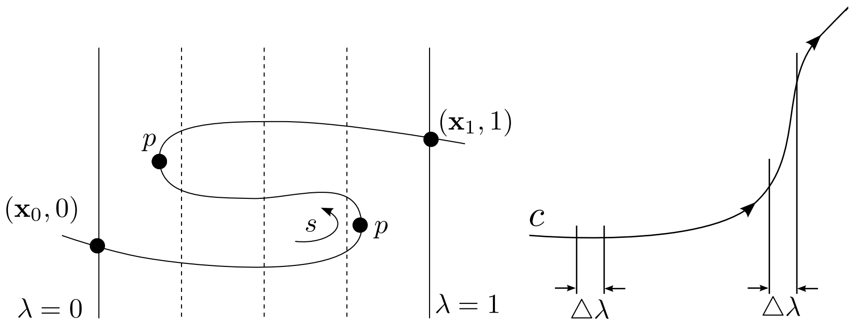
A point where D$\mathcal{H}$ has full rank is said to be a *regular point* of $\mathcal{H}$. If the inverse image of a point $y \in \mathbb{R}^n$ contains only regular points, then y is said to be a *regular value* of $\mathcal{H}$. On the other hand, if at a point every $n \times n$ minor of D$\mathcal{H}$ is singular, that point is called a *critical point* of $\mathcal{H}$. If the inverse image of y has at least one critical point of $\mathcal{H}$, y is said to be a *critical value* of $\mathcal{H}$.

If all zero points of $\mathcal{H}$ are regular points, then this curve of zeros is diffeomorphic to a circle or the real line. Allgower and Georg [3] discuss the existence of such a curve of zeros. It is generally sufficient to require some boundary condition that prevents the curve from running to infinity before intersecting the level $\lambda = 1$ or from returning back to level $\lambda = 0$. Such boundary conditions are generally specific to the problem at hand.

In some cases, a parametrization of the curve of zeros with respect to λ will fail at the turning points of the curve with respect to λ as shown in Figure 3.3(a). While in other cases, the λ parametrization will require one to choose extremely small $\triangle\lambda$ increments in order to trace the curve c (see Figure 3.3(b)). One possible remedy in this case is to use the arc length s as the parameter for the curve.

3.2.1 Bifurcation points

The curve of zeros $c(s)$ might contain bifurcation points. Some of the fundamental work of bifurcation theory and the numerical solution of bifurcation problems are due to [68, 67, 69]. Numerical bifurcation theory and computer programs can be, for example, found in [114]. Formally a bifurcation point can be defined as follows [3]



(a) Parametrization failure at turning points p. (b) λ step length requirement.

Figure 3.3: *Drawbacks of λ parametrization.*

Definition 3.6 For an arbitrarily smooth $\mathcal{H} : \mathbb{R}^{n+1} \mapsto \mathbb{R}^n$, suppose that $c : I \mapsto \mathbb{R}^{n+1}$ is a smooth curve defined over an open interval I containing zero, and parametrized with respect to arc length such that $\mathcal{H}(c(s)) = 0$ for $s \in I$. The point $c(0)$ is called a *bifurcation point* of $\mathcal{H} = 0$ if there exists an $\epsilon > 0$ such that every neighborhood of $c(0)$ contains zero-points z of $\mathcal{H}$ which are not in $]-\epsilon, \epsilon[$ (see Figure 3.4).

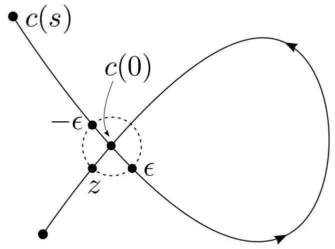


Figure 3.4: *Bifurcation point.*

A bifurcation point of $\mathcal{H} = 0$ must be a critical point of $\mathcal{H}$. Allgower and Georg [3] discuss handling of simple bifurcation points and branch switching while tracing the curve of zeros. In this work, the bifurcation points are handled in a different fashion. Homotopies are computed in sub-intervals of the domain such that each sub-interval contains only one branch that is part of the curve of zeros (see Chapter 7).

3.2.2 Numerical curve tracing

The assumption behind numerical curve tracing for a smooth map $\mathcal{H} : \mathbb{R}^{n+1} \mapsto \mathbb{R}^n$ is the existence of a point $\hat{\mathbf{x}}_0 \in \mathbb{R}^{n+1}$ such that

1. $\mathcal{H}(\hat{\mathbf{x}}_0) = 0$, and
2. The Jacobian matrix $\mathcal{H}'(\hat{\mathbf{x}}_0)$ has maximal rank, which is n.

Under these assumptions, there exists a smooth curve $c(\alpha) : I \mapsto \mathbb{R}^{n+1}$ for some open interval I containing zero such that for all $\alpha \in I$[3]

1. $c(0) = \hat{\mathbf{x}}_0$, the initial condition is met,
2. $\mathcal{H}(c(\alpha)) = 0$, the curve is in the kernel of $\mathcal{H}$,
3. $\text{rank}\,(\mathcal{H}'(c(\alpha))) = n$, all points on the curve are regular points, and
4. $c'(\alpha) \neq 0$, $c(\alpha)$ is a monotonic curve such that the parametrization α does not fail.

Two methods for numerically tracing c are discussed next.

Predictor-corrector (PC) methods

Predictor-Corrector methods numerically trace the curve c by generating a sequence of points $\hat{\mathbf{x}}_i, i = 1, 2, \cdots$ along the curve satisfying some tolerance condition like $||\mathcal{H}(\hat{\mathbf{x}}_i)|| \leq \epsilon$ for some $\epsilon > 0$. The starting point for the sequence is $\hat{\mathbf{x}}_0 \in \mathbb{R}^{n+1}$ such that $\mathcal{H}(\hat{\mathbf{x}}_0) = 0$.

A new point $\hat{\mathbf{x}}_{i+1}$ can be generated along the curve using a *predictor step* followed by a *corrector step*. A predictor step is generally a suitable numerical integration scheme. Most commonly employed predictor step is the *Euler predictor*

$$\hat{\mathbf{y}}_{i+1} = \hat{\mathbf{x}}_i + h\mathbf{t}(\mathcal{H}'(\hat{\mathbf{x}}_i)), \tag{3.7}$$

where $h > 0$ represents the step size and $\mathbf{t}(\mathcal{H}'(\hat{\mathbf{x}}_i))$ is the tangent vector induced by $\mathcal{H}'(\hat{\mathbf{x}}_i)$.

A powerful corrector step is available due to the fact that $\mathcal{H}(\hat{\mathbf{x}}) = 0$. Even with a poor predictor point $\hat{\mathbf{y}}_{i+1}$, an iterative corrector process will show rapid convergence. Let $\hat{\mathbf{z}}_{i+1}$ denotes the point on c nearest to $\hat{\mathbf{y}}_{i+1}$ (see Figure 3.5). The point $\hat{\mathbf{z}}_{i+1}$ solves the following optimization problem

$$||\hat{\mathbf{z}}_{i+1} - \hat{\mathbf{y}}_{i+1}|| = \min_{\mathcal{H}(\hat{\mathbf{z}})=0} ||\hat{\mathbf{z}} - \hat{\mathbf{y}}_{i+1}||. \tag{3.8}$$

A Newton-like method can be used to arrive at the desired solution point $\hat{\mathbf{x}}_{i+1}$ in a few iterations.

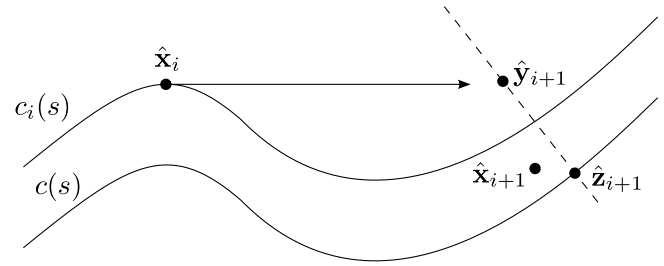


Figure 3.5: *Predictor point* $\hat{\mathbf{y}}_{i+1}$ *and corrector point* $\hat{\mathbf{x}}_{i+1}$.

Piecewise-linear (PL) methods

A Piecewise-Linear method proceeds by following a piecewise-linear curve c_τ that approximates c over a *triangulation* τ of $\mathbb{R}^{n+1}$. A triangulation τ of $\mathbb{R}^{n+1}$ is a subdivision of $\mathbb{R}^{n+1}$ into $(n+1)$-simplices such that

1. any two simplices in τ either share a face or nothing,
2. any bounded set in $\mathbb{R}^{n+1}$ intersects only finitely many simplices in τ.

In order to trace c_τ, a piecewise linear approximation $\mathcal{H}_\tau$ of $\mathcal{H}$ is defined by

1. $\mathcal{H}_\tau(v) = \mathcal{H}(v)$ for all vertices v of τ,
2. for any $(n+1)$-simplex $\sigma = [v_1, v_2, \cdots, v_{n+2}] \in \tau$, the restriction $\mathcal{H}_\tau|_\sigma$ of $\mathcal{H}_\tau$ to σ is an affine map.

Since, $\mathcal{H}_\tau$ is affine,

$$\mathcal{H}_\tau(\hat{\mathbf{x}}) = \mathcal{H}\left(\sum_{i=1}^{n+2} \alpha_i v_i\right) = \sum_{i=1}^{n+2} \alpha_i \mathcal{H}(v_i), \tag{3.9}$$

where $\alpha_i, i = [1, n+2]$ are barycentric coordinates of $\hat{\mathbf{x}}$ with respect to the simplex σ in which it lies. The set $\mathcal{H}_{\mathcal{T}}^{-1}(0)$ contains a polygonal path $c_{\mathcal{T}} : \mathbb{R} \mapsto \mathbb{R}^{n+1}$ that approximates c. Such a path is traced via methods of linear programming such as the simplex method. Figure 3.6 illustrates the basic idea behind piecewise-linear path following.

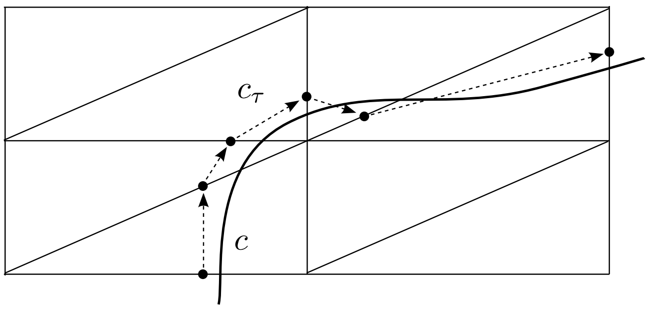


Figure 3.6: *Piecewise-Linear path following.*

Multivariate polynomial system of equations arrive frequently in many problems. Starting in 1970's, a homotopy continuation based approach to solving such systems called *polynomial continuation* was developed. By using algebraic geometry and homotopy theory, special algorithms can be designed to solve polynomial systems in a theoretically complete and practically robust manner. Besides being general, polynomial continuation has the advantage that very little symbolic information is required from the polynomial system [130].

There exist libraries for numerical computation of homotopy and solution of system of non-linear equations. HOMPACK [141] is a suite of globally convergent homotopies for solving nonlinear systems of equations. It includes subroutines for fixed point, zero finding, and general curve tracking problems. PHC [136] implements homotopy continuation methods to compute approximations to all isolated solutions of a system of n polynomial equations in n unknowns. It offers many root counting methods including Bézout number, multihomogenous Bézout number, and mixed volume. Bézout's theorem is an important and classical theorem in algebraic geometry. It states that two algebraic curves of degrees m and n, with no common components, will share mn points, when counted with multiplicity. Thus, *Bézout number* accounts for the number of solutions to a system of polynomial equations, which is equal to the total degree of the system. The *multihomogenous Bézout number* is an upper bound on the number of geometrically isolated solutions in a product of projective spaces [94]. The *mixed volume* of a collection of m polytopes $\{P_i\}$ and a set of non-negative real parameters λ_i is the coefficient of $\lambda_1 \cdots \lambda_m$ in the volume of the Minkowski sum $\sum \lambda_i P_i$, which is a homogeneous polynomial in λ_i [132]. The mixed volume of Newton polytopes gives a sharp upper bound for the number of complex isolated solutions with nonzero coefficients. PhoM [56] is a library for a polyhedral homotopy continuation method for finding all isolated solutions to a non-linear system. HomLab [130] is a Matlab toolbox for polynomial continuation.

3.3 Davidenko's equation

Under various conditions, $\mathcal{H}(\mathbf{x}, \lambda)$ will have a zero for each λ in $[0, 1]$, and the zero $\mathbf{x}_0$ of $\mathcal{H}(\mathbf{x}, 0)$ may lead to an approximation to a zero of $\mathcal{H}(\mathbf{x}, 1)$ via the *curve of zeros* $\mathbf{x}(\lambda)$ that satisfies $\mathcal{H}(\mathbf{x}(\lambda), \lambda) \equiv 0$ with $\mathbf{x}(0) = \mathbf{x}_0$ [93].

Besides the methods discussed above, the curve of zeros $\mathbf{x}(\lambda)$ can be determined by formulating an initial value problem. Considering the homotopy $\mathcal{H}(\mathbf{x}, \lambda) : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}^n$ as the set of n non-linear algebraic equations, we assume the existence of the curve of zeros $\mathbf{x}(\lambda)$, such that $\mathbf{x}(0) = \mathbf{x}_0$. Differentiating $\mathcal{H}(\mathbf{x}(\lambda), \lambda) = 0$ with respect to the homotopy parameter λ, we get

$$\mathbf{J}\frac{\mathrm{d}\mathbf{x}(\lambda)}{\mathrm{d}\lambda} + \frac{\partial \mathcal{H}(\mathbf{x}(\lambda), \lambda)}{\partial \lambda} = 0, \tag{3.10}$$

where $\mathbf{J}$ is the Jacobian of the terminal map. For a homotopy defined as

$$\mathcal{H}(\mathbf{x}(\lambda), \lambda) = f(\mathbf{x}(\lambda)) + (\lambda - 1)f(\mathbf{x}_0), \tag{3.11}$$

where $f = 0, f : \mathbb{R}^n \mapsto \mathbb{R}^n$ is the non-linear system whose solution is required,

$$\frac{\partial \mathcal{H}(\mathbf{x}(\lambda), \lambda)}{\partial \lambda} = f(\mathbf{x}_0). \tag{3.12}$$

Therefore, (3.10) gives

$$\frac{\mathrm{d}\mathbf{x}(\lambda)}{\mathrm{d}\lambda} = -\mathbf{J}^{-1} f(\mathbf{x}_0). \tag{3.13}$$

The ordinary differential equation (3.13) along with the initial condition $f(\mathbf{x}_0) = 0$ is known as *Davidenko's equation* [36].

An approximate solution to (3.13) at $\lambda = 1$ can be taken as a root of $f(\mathbf{x}) = 0$:

$$f(\mathbf{x}(1)) = f(\mathbf{x}_0) + \int_0^1 \mathbf{J}\frac{\mathrm{d}\mathbf{x}(\lambda)}{\mathrm{d}\lambda}\mathrm{d}\lambda. \tag{3.14}$$

Davidenko's method consists of finding an approximate solution to the initial value problem at $\lambda = 1$ by using appropriate numerical integration techniques

[109, 50, 93]. By considering the approximate solution of (3.13) as initial guess, it becomes easy to apply a Newton-type methods for computing accurate solution to $\mathcal{H} = 0$ [3]. Computationally, solution to Davidenko's equation is much faster than other methods of tracing c [142].

3.4 Homotopy formulation for reconstruction from acoustic signals

Wayburn and Seader [142] discuss some objections on using numerical homotopy based schemes and present viable solutions to them. In formulating homotopies for reconstruction from acoustic images, emphasis is given on solvable systems that do not impose large computational burden, and still providing a reasonably good reconstruction. The reconstruction problem presented in this work is slightly different than the one presented in section 3.2. Here, the initial map and the terminal maps of the homotopy have known solutions, and a reasonable path connecting them is sought after (see Figure 3.7).

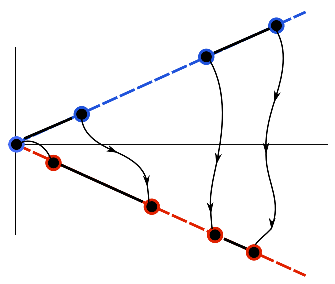


Figure 3.7: *Path connecting two sonar beams maps.*

More details on the design of homotopies based on the geometry of the sonar beams is presented in chapter 7. Homotopies satisfying smoothness and monotonicity are also designed incrementally starting from the convex homotopy. Homotopies discussed in chapter 7 are designed specifically to take advantage of the geometry of sonar beams and take advantage of the inherent ordering within the beams while parameterizing the resulting surface in $\mathbb{R}^3$. Figure 3.8 shows the arrangement of a 2D fan of sonar beams.

The algorithm developed for reconstruction from sonar beams is generalized to reconstruction from arbitrary cross sections in chapters 8 and 9. The developed homotopy does not assume existence of any ordering within the cross sections.

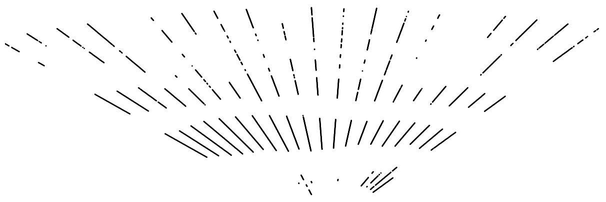


Figure 3.8: *Arrangement within beams of a 2D fan.*

Such problems can arise in situations when an object is scanned (using range scanners like single beam echosounders) from different directions. The presented algorithm is generic in nature and results are shown in $\mathbb{R}^2$. Figure 3.9 illustrates the problem setup.

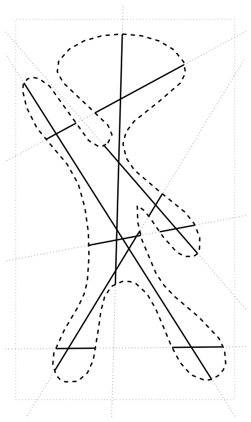


Figure 3.9: *Reconstruction problem with arbitrary cutting lines.*

Chapter 4

Object segmentation and reconstruction

Previous chapters discussed basics of underwater acoustics and presented the mathematical background of homotopy continuation and available numerical tools. The purpose of this chapter is to bridge the gap between presented domain specific background and mathematical preliminaries from the previous chapters and the reconstruction approach presented in the following chapters (originally written as publication manuscripts).

Scientific literature on image segmentation is very rich and an overwhelming number of publications suggest the level of active research in this field. Acoustic images suffer from speckle noise (see Chapter 2, subsection 2.4.4). Speckle noise is a phenomenon caused when a coherent imaging system is used to image a surface that is rough on the scale of the wavelength used. Each surface element produces reflections in every resolution cell that combine to form an interference pattern by adding up either constructively or destructively resulting in a speckle pattern.

A number of speckle suppression techniques have been developed over the years to reduce noise in acoustic images. A general approach is to smooth the speckle in images. Many speckle filters are adaptive to the local texture information. At the very basic level, median filtering works very well to suppress impulse noise while retaining sharp edges. Some of the high performance filters are

Lee [78], Kuan[75], Frost [46], modified Lee, and modified Frost filters [84]. Speckle filters based on the maximum a posteriori (MAP) probability [84, 64] and wavelet suppression [41, 59] are also considered good. Another class of speckle reducing methods is based on minimization of total variation (L_1 norm of the gradient) of the acoustic image [111, 40]. Krissian et al. [73] provide an improvement over the classical anisotropic diffusion [73] on ultrasound images constrained by the speckle noise model. Each speckle reducing filter presents trade-offs between noise reduction, image smoothing, edge preservation, and computational complexity.

An image segmentation algorithm aims at partitioning an image into regions that are non-overlapping, uniform and homogeneous with respect to some characteristic derived from the image intensity and textural variation. The boundaries of each segment should be simple, with minimum holes, and spatially accurate. In general, it is difficult to fulfill all the criteria at the same time. With acoustic images, the boundary between the object and the background is not very well defined. The presence of fully developed speckle pattern makes it harder to get a reasonable segmentation. The simplest segmentation method still in use in fisheries is thresholding [126, 112]. A threshold is generally selected based on the target strength of the fish seen in the images. This usually serves the purpose of removing the background noise, but in presence of air bubbles in water, a simple threshold can give misleading results. Furthermore, the boundary of segmented objects is not very accurate with thresholding. Markov Random Fields have been shown to segment sonar images in an unsupervised fashion [91, 92].

Derin and Elliott [39] estimate a noise model by an unsupervised iterative estimation procedure based on the Iterative Conditional Estimation (ICE). Methods of data clustering like k-means[105] have been successfully applied to image classification. Classical clustering is applied to images by using Gibbs random fields. Mean-shift based clustering in color images has been popularized by Comaniciu and Meer [30, 31]. Mean-shift is a non-parametric kernel density estimation method that seeks the mode of the data by an iterative and converging procedure. Acoustic range image segmentation using mean-shift clustering in 7-dimensional feature space has been demonstrated in [23]. Partial differential equation (PDE) based segmentation methods, specially the level set method introduced by Sethian [113], and Malladi and Sethian [87], have found tremendous success in medical image and volume segmentation [27, 137, 9, 79, 122]. Level set methods can be applied in a straightforward fashion to sonar images [82, 123].

In the current reconstruction approach, a collection of noise removal techniques and the level set method have been applied to get a segmentation of the sonar imagery. This segmentation is required for the homotopy based reconstruction algorithms. The general methodology adopted in this work is outlined in Figure 4.1.

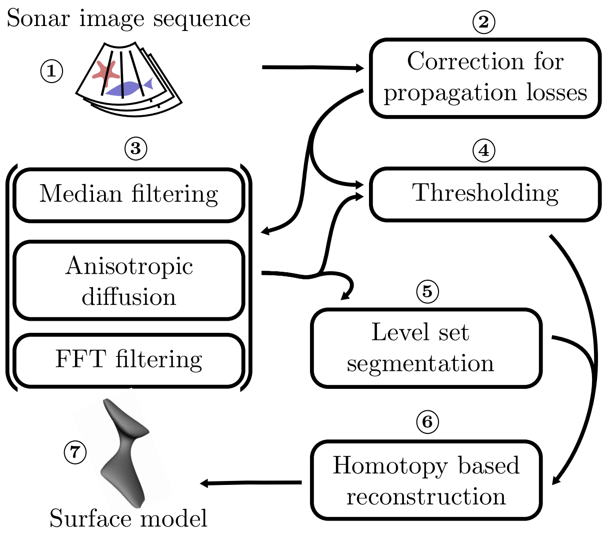


Figure 4.1: *Reconstruction method. Parts of the method between ② and ⑥ are a set of optional processing steps, in which one of the steps ④ and ⑤ is necessary to get a segmented volume for input to ⑥, while step ③ is completely optional.*

The size of the volumetric dataset considered in this work is of the order of 100 million voxels, therefore parallel computing technologies are explored for near real time performance of the reconstruction method. Before delving into the main methodologies, a brief overview of the parallel computing technologies used in this work are outlined in the next section.

4.1 Parallel computing

High computational needs and (near) real time performance are quite common in scientific computation tasks. Parallelism is a way of speeding up computations at the cost of high computing resources. Multi-core architectures have started a new era of computing and boosted performance and efficiency of parallel programs. One of the most successful in programming parallel computers is the Message Passing Interface (MPI) [45], which conforms to the message-passing model. The implementations contain parallelization functions with language interfaces to Fortran and C/C++. While there are lot of parallel computing tools around, the two parallel computing technologies used in this work are OpenMP and CUDA.

The most commonly used approach for shared-memory parallelization is the Open Multi-Processing (OpenMP) [35] application programming interface (API). The target languages for OpenMP are Fortran and C/C++. OpenMP essentially consists of a set of compiler directives that are used to describe parallelism in the source code, and a small library of supporting routines. The compiler in use must be OpenMP enabled to use the directives and the library. Simplicity in programming is a definite advantage with OpenMP as compared to Pthreads or MPI.

Another mature technology providing massive parallelism by using the graphics card or the Graphics Processing Unit (GPU) is CUDA (Compute Unified Device Architecture) from Nvidia. The programmable GPU is a highly parallel, multithreaded, many-core processor with high computational power and good memory bandwidth [97]. The GPU was designed for graphics rendering that involved tasks with high data parallelism. Therefore, a GPU can be utilized for data-parallel computations. CUDA is a parallel programming model and instruction set architecture.

4.1.1 OpenMP

OpenMP is a set of compiler directives and callable runtime library routines that extend Fortran and C/C++ to express shared-memory parallelism [35, 28]. It follows the *fork/join model* of execution, thus spawning multiple threads to execute parallel sections of the code and joining the result to pass control to the master thread at the end of the parallel section. The slave threads are created when the program first encounters a parallel section. These additional threads are not destroyed, but saved for future use in either suspended or slept state.

OpenMP's constructs fall into five categories

1. **Parallel regions**
   Threads are created with the `omp parallel` pragma in C/C++. The body of a loop after the pragma defines a parallel region.

```
double Data[10000];
omp_set_num_threads(8);
#pragma omp parallel {
  int ID = omp_get_thread_num();
  task(ID,Data);
}
```

2. **Work sharing**
   The `omp for` work sharing construct splits up loop iterations among the threads in a parallel region. A shortcut for specifying both a parallel region and the work sharing `omp for` construct is `omp parallel for`.

```
#pragma omp parallel
#pragma omp for
for (int i=0; i<10000; i++) {
  task(Data[i]);
}
```

   A `for` construct cannot be used for a reduction operation like a summation. OpenMP provides the construct `omp reduce (op:list)`, where `op` is the reduction operator, and `list` is a list of reduction variables. `omp sections` is another work-sharing construct that assigns different jobs to each thread in a parallel region. The thread distinction can be made depending on the thread id obtained via OpenMP call to `omp_get_thread_num()`.

3. **Data environment**
   In the shared-memory programming model, global variables are shared among threads (file scope and static variables in C/C++). Automatic variables within a statement block are private. Stack variables in routines called from parallel regions are also private. A private variable is a private copy of the variable for a thread. Constructs and clauses available to selectively change storage attributes are: `shared`, `private`, `firstprivate`, and `lastprivate`.

4. **Synchronization**
   Constructs available for synchronization are:

   - the `critical` and `end critical` constructs, which define a *critical section* where only one thread can enter at a time;
   - the `atomic` construct, which defines a critical section that only contains one simple statement;
   - the `barrier` construct, which makes the threads wait until all of them arrive, and is usually implicit (e.g. at the end of `for` construct);
   - the `ordered` construct, which enforces the sequential order for a block of code;
   - the `master` construct, which marks a block of code to be executed only by the master thread, while other threads just skip it;

- the `single` construct, which marks a block of code to be executed only by one thread; and
- the `flush` construct, which denotes a sequence point where a thread tries to create a consistent view of memory.

5. **Runtime routines and environment variables** The library routines and environment variables in OpenMP control and retrieve the state of the system. The number of threads can be controlled using the routines `omp_set_num_threads()`, `omp_get_num_threads()`, `omp_get_thread_num()`, and `omp_get_max_threads()`. When not explicitly mentioned in the code, OpenMP tries to spawn as many threads as there are number of CPU cores present. The number of processors in the system can be retrieved by a call to `omp_num_procs()`.

4.1.2 CUDA

CUDA stands for Compute Unified Device Architecture. It is a general purpose parallel computing architecture with a new parallel programming model and instruction set architecture that leverages the parallel computing engine in Nvidia GPUs to solve many complex computational problems in a more efficient way than on a CPU [97].

Hardware configuration

A GPU (referred to as a *device*) is a set of Streaming Multiprocessors (SMs) employing a new SIMT (Single-Instruction Multiple-Threads) multi-processors computational architecture. Each SM consists of eight Scalar Processor (SP) cores. Every SP has it's own registers while every SM has access to small (16 KB) but high speed *shared memory*. The whole device has access to *device memory*, *constant memory*, and *texture memory*. The constant and texture memory are cached while the device memory is not. In fact, the shared memory can be views as a user managed cache.

Thread configuration

CUDA exposes a C like programming environment with a minimal set of extensions and a few C++ like features. CUDA code executes on the GPU in the form of functions called *kernels*, which get called N times in parallel by

N different CUDA threads. CUDA threads are lightweight GPU threads with minimal creation overhead as opposed to CPU threads. Thousands or millions of threads are required to achieve full efficiency of the parallelism. The minimal set of extensions to the the C language include:

- Function type qualifiers:

  ```
  __device__ <return_type> device_func(...);
  __global__ void kernel_func(...);
  __host__ <return_type> host_func(...);
  ```

- Variable type qualifiers:

  ```
  __device__ <data_type> device_var;
  __constant__ <data_type> constant_var;
  __shared__ <data_type> shared_var;
  ```

- Directive for kernel execution:

  ```
  kernel_func<<<gridDim, blockDim>>>(...);
  ```

- Built-in variables

  ```
  gridDim (dim3)
  blockDim (dim3)
  blockIdx (uint3)
  threadIdx (uint3)
  warpSize (int)
  ```

Further, the runtime library is split into:

- A host component
- A device component
- A common component

The CUDA C compiler is called *nvcc*. The set of threads in a kernel launch are hierarchically grouped in a *block* and the blocks are grouped in a *grid*. The threads in a block can communicate using the shared memory and basic thread synchronization (using `__syncthreads()`) is possible. No intercommunication and no barrier synchronization is possible between threads across different blocks. All threads have access to the global memory, which has a higher delay in access.

A kernel is defined using the `__global__` declaration specifier and the number of CUDA threads for each call is specified by the `<<<...>>>` syntax [97].

```
// Kernel definition
__global__ void vecAdd(float* A, float* B, float* C)
{
  int i = threadIdx.x;
  C[i] = A[i] + B[i];
}

int main()
{
  // Kernel invocation
  vecAdd<<<1, N>>>(A, B, C);}
}
```

Each thread has a unique *thread ID* accessible from within the kernel using the `threadIdx` variable.

With this brief introduction to the parallel computing technologies, the disusion is now turned to processing of raw acoustic images.

4.2 Correction for propagation losses

The first step in acoustic data processing is to correct the images to compensate for various losses incurred by a sound wave while traveling underwater. This is mostly a standard procedure and converts the raw signal into meaningful values (see Chapter 2). The raw signal power P is converted to an S_v value using [127, 1]

$$\mathrm{S}_v = \mathrm{P} + 20\log_{10}(\mathbf{r}) + 2\mathrm{C}_{abs}\mathbf{r} - \mathrm{S}_v^{const} \tag{4.1}$$

where S_v is the volume backscattering strength in dB, P is the power in dB relative to 1 Watt, $\mathbf{r}$ is the range vector in meters relative to 1 m, C_{abs} is the absorption coefficient in dBm^{-1}, the constant S_v^{const} is calculated as

$$\mathrm{S}_v^{const} = 10\log_{10}\left(\frac{\mathrm{P}_t\lambda^2 v\tau}{32\pi^2}\right) + 2\mathrm{G} + \psi \tag{4.2}$$

where P_t is the transmit power in watts, λ is the wavelength of pulse in meters, v is the velocity of sound in water in ms^{-1}, τ is the pulse length in seconds, G is the applied gain in dB of the transceiver including the effects of transducer energy conversion efficiency and receiver gain, and ψ is the equivalent beam angle in dB. The MS70 Simrad dataset is compensated for absorption and spreading losses in this way.

The s7k dataset from RESON is already compensated for such losses and therefore doesn't need to be corrected. However, the raw values from the device output are encoded in 16 bit unsigned integers. These values are relative backscatter values and can be converted into the more familiar dB units by taking the logarithm and then adding a constant term coming from the backscatter strength of a copper sphere obtained via a sonar calibration process. The expression looks like

$$\mathrm{TS} = 20\log_{10}(\mathrm{P}) + \mathrm{TS}_{\mathrm{Cu}} - 20\log_{10}\left(\max\left(\mathrm{P}_{\mathrm{Cu}}\right)\right), \tag{4.3}$$

where TS is the target strength of the objects in dB, P is the relative pressure value, $\mathrm{TS}_{\mathrm{Cu}}$ is the standard copper sphere target strength at 400 KHz frequency pulse, and $\max\left(\mathrm{P}_{\mathrm{Cu}}\right)$ is the maximum relative pressure value of the copper sphere obtained during a calibration process.

4.3 Image denoising

Noise comes from unwanted signals that are present in the medium but are independent of the echosounder transmission. Various artifacts also arise due to the echosounder and the process of beamforming itself. The main types of noise (see Figure 4.2) seen in the images are:

- the *Speckle pattern*, that is generated by the interference of multiple scatterers present in the insonified volume. In addition to the useful signal, the resulting signal also contains random intensity variations resulting from combinations of many waves with different phases.

- the *Transmit pulse echo*, that is appearing immediately after a pulse is transmitted into the water. The noise level quickly reduces to the thermal noise level.

- the *Sidelobe intensities*, that are visible for strong scatterers. The sidelobes cannot be completely eliminated (see Chapter 2) and therefore even though the intensities are small compared to the main-lobe intensity, these become visible for strong reflections [86, Chap. 5].

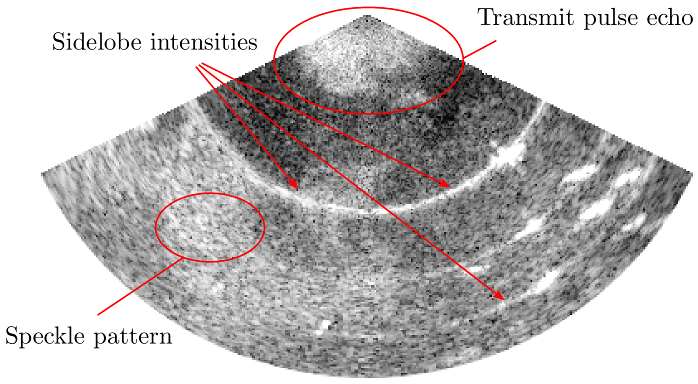


Figure 4.2: *Various noises in an acoustic image.*

4.3.1 Speckle pattern

A simple strategy to suppress speckle noise is to perform median filtering over the image. Since the size of the resolution cell increases while moving away from the transducer, the size of the speckles also increases. It is therefore better to perform filtering operations in the untransformed domain (i.e., in the Cartesian beam space) (see Figure 4.3(a)). This has an added advantage of processing smaller images since the rectangular to polar transformation on acoustic images generally results in an expansion of the image size. A median filter of size $[3 \times 3]$ is sufficient for one pixel sized speckles (see Figure 4.3(b)). A larger window size generally results in excessive smoothing.

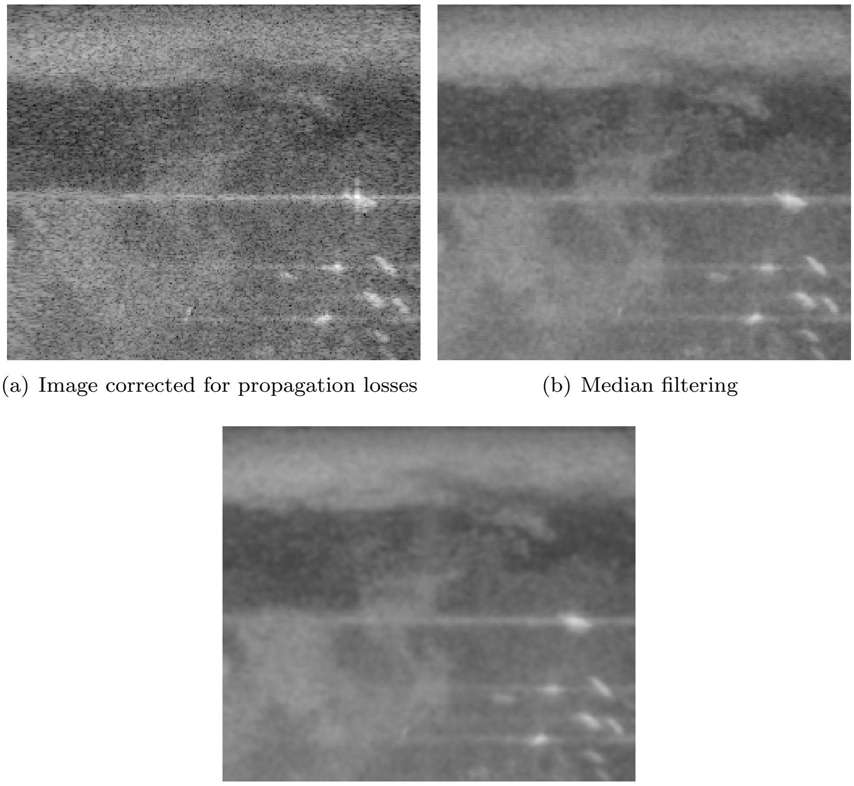
After the median filtering step, the image still contains larger sized speckles. Since the speckle intensity is lower than that of the object intensities, a smoothing step can smear out much of the speckle into the background. Anisotropic filtering by Perona and Malik [106] performs better at preserving the boundaries while smoothing the image. The anisotropic diffusion equation is given by

$$I_t = \text{div}\left(c(x,y,t)\nabla I\right) = c(x,y,t)\Delta I + \nabla c \cdot \nabla I, \tag{4.4}$$

where c is the conduction coefficient, div is the divergence operator, Δ is the Laplace operator, and ∇ is the gradient operator. Perona and Malik show how a suitable choice of c can result in selective smoothing that preserves the edges in an image. For instance, c could be a function of the edges of the image

$$c(x,y,t) = g\left(||\nabla I(x,y,t)||\right), \tag{4.5}$$

where $||\nabla I(x,y,t)||$ is an edge estimate of the image I and $g(\cdot)$ is a monotonically decreasing function. The result of anisotropic diffusion applied on the median filtered image is shown in Figure 4.3(c).



(a) Image corrected for propagation losses

(b) Median filtering

(c) Anisotropic diffusion

Figure 4.3: *Acoustic image despeckling.*

4.3.2 Transmit pulse echo

The transmit pulse echo is a result of decaying oscillations of the transducer after the main pulse is transmitted into the medium. The maximum range for transmit pulse can be calculated as

$$R_{max}^{tx} = \frac{n\tau c}{2k}, \tag{4.6}$$

where n is the number of sectors, τ is the pulse duration, c is the speed of sound in water measured at the depth where observation is made, and k is the number of pulses in one transmission. A safety margin is generally added to this value. Signal in this range is discarded from the initial part of the signal. The signal in the near-field does not contain much useful information so discarding the initial part of the signal is a usual practice.

4.3.3 Sidelobe intensities

The sidelobe intensities in a polar transformed acoustic image are seen as horizontal streaks in the corresponding beam space image (see Figure 4.3(a)). Every beam has a beam pattern with a main lobe (most sensitive) in the pointing direction of the beam and some sensitivity in other directions due to finite size of the sonar aperture. These other side lobes which transmit and receive energy from other directions than the main pointing direction will transmit and receive some intensity from the direction of stronger echoes. These echoes will be displayed as weaker echoes since the transmit and receive sensitivity in the other directions are smaller than in the main beam pointing direction. Such an artifact could be difficult to get rid of in the spatial domain, therefore a frequency domain denoising approach is adopted here. Let's first take a look at the Fourier spectrum shown in Figure 4.4(a) of the anisotropic diffusion filtered image. The spectrum clearly shows presence of strong vertical lines corresponding to horizontal linear features in the original image. A notch filter [52] can be designed to eliminate frequencies that emphasize the horizontal streaks. A modified filter used here is a pair of vertically symmetric closed polygons enclosing the bright vertical lines in the spectrum (see Figure 4.4)(b). In order to avoid ringing artifact in the inverse transformed image, a smoothed filter is used. Smoothing is enforced via the following function applied to the distance transform of the binary mask resulting from the polygons.

$$f_\epsilon(z) = \frac{1}{2}\left(1 + \frac{2}{\pi}\tan^{-1}\left(\frac{z}{\epsilon}\right)\right), \tag{4.7}$$

where ϵ is a regularizing parameter. An inverse Fourier transform of the multiplication of the regularized filter with the Fourier transform of the image results in the denoised image shown in Figure 4.5(a). Note that the horizontal streaks have disappeared. One should be careful while designing the notch filter to not remove too much of the salient details from the original image. Some of the image power is lost during Fourier denoising but in our case it is considerably low. The image in Figure 4.5(b) shows the difference between the input to the FFT filtering and the filtered image. The highest intensity in the residual image amounts to only 7.18% of the highest intensity of the input image.

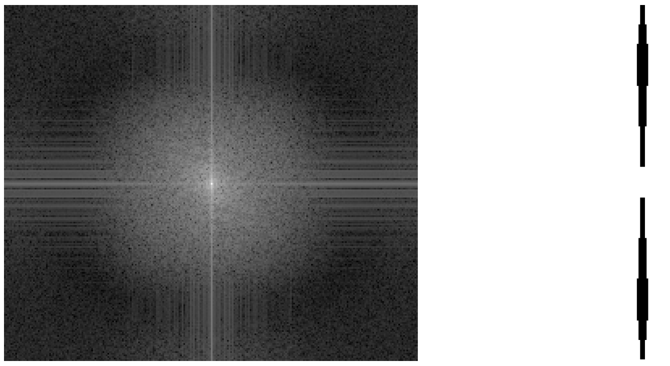

(a) Logarithm of Fourier spectrum (b) Filter mask

Figure 4.4: *Power spectrum and filter design.*

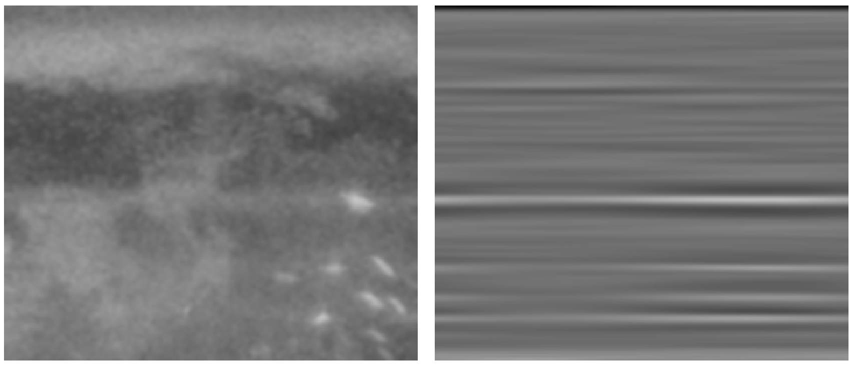

(a) Filtered image (b) Residuals (with scaled intensities)

Figure 4.5: *FFT filtering for streak removal.*

4.4 Level set segmentation

The basic idea behind level set segmentation is to start with a closed curve in the domain of the image. This curve then deforms under the influence of various forces and finally takes the shape of the desired objects. Topology changes of the curve are permitted during the evolution. Therefore, the curve can break into multiple curves and multiple curves can combine together.

The deformable curve that permits topological changes can be represented in the most general way as a level set of an implicit function. The driving force is generally composed of a curvature energy and an image energy. Curvature energy tends to move the curve under the influence of its curvature via the *mean curvature flow* [101, chap. 4]. Such a force makes the curve deform into a circle and eventually collapse into a point before disappearing. It is the image energy that prevents the curve from vanishing. Such an energy stops the curve at the required location determined based on the intensity of the image. Various energy formulations are based on image edges (image gradients) [88], and the Mumford-Shah model [26].

A CUDA based level set solver suited for segmentation of sonar images is discussed in Chapter 5 [115]. It uses the modified Mumford-Shah level set formulation by Chan and Vese [26] coupled with a noise suppression scheme to segment fishes from large acoustic datasets. The presented scheme works on raw acoustic values. An extension to this work is presented in Chapter 6 [122] (also submitted for publication in [123]), where the CUDA solver is extended to a Multi-phase and higher order level set scheme.

4.5 Homotopy reconstruction

A brief overview of the use of continuous deformations in object reconstruction is shown in Chapter 7. The simplest homotopy reconstruction algorithm (the linear homotopy) takes as input, information along two beams at a time. The information along a beam is encoded in its characteristic function. Here, the characteristic function is a piecewise-constant segmentation of the original signal such that the background has the zero value while the object has the one value along the beam. A suitable beam function is constructed from such a signal mass. These beam functions are used to compose a cascade of homotopies that are smooth at the boundaries. The zero level set of the homotopies constitutes the surface of the reconstruction. Chapter 7 [117] contains a detailed description of the homotopy reconstruction method and various different homotopies for a smooth reconstruction.

An interesting extension of this reconstruction problem is presented in Chapter 8 [118] where the problem of arbitrary cutting lines is presented and a solution to reconstruction using multiple variable homotopy is suggested. Results are shown for a 2D case and the proof of smoothness of reconstruction is included in Appendix A. A variant of this method using Voronoi diagram based edge barycentric coordinates for polygons is presented in Chapter 9 [116].

Chapter 5

CUDA based level set method for 3D reconstruction of fishes from large acoustic data

O. Sharma[†], and F. Anton[†]

[†]Department of Informatics and Mathematical Modelling,
Technical University of Denmark, Lyngby, 2800, Denmark

Abstract

Acoustic images present views of underwater dynamics, even in high depths. With multi-beam echo sounders (SONARs), it is possible to capture series of 2D high resolution acoustic images. 3D reconstruction of the water column and subsequent estimation of fish abundance and fish species identification is highly desirable for planning sustainable fisheries. Main hurdles in analyzing acoustic images are the presence of speckle noise and the vast amount of acoustic data. This paper presents a level set formulation for simultaneous fish reconstruction and noise suppression from raw acoustic images. Despite the presence of speckle noise blobs, actual fish intensity values can be distinguished by extremely high values, varying exponentially from the background. Edge detection generally gives excessive false edges that are not reliable. Our approach to reconstruction is based on level set evolution using Mumford-Shah segmentation functional that does not depend on edges in an image. We use the implicit function in conjunction with the image to robustly estimate a threshold for suppressing noise in the image by solving a second differential equation. We provide details of our estimation of suppressing threshold and show its convergence as the evolution proceeds. We also present a GPU based streaming computation of the method using Nvidia's CUDA framework to handle large volume data-sets. Our implementation is optimized for memory usage to handle large volumes.

5.1 Introduction

One of the areas of interest in fisheries research is to reconstruct moving schools of fishes in a water column. Presence of strong speckle noise is a major problem in segmenting acoustic images. This makes selection of a threshold for binary segmentation very difficult [108]. Multibeam echo sounders are not outdated by hyperspectral underwater imagers, but they just complement very well. Acoustic sensors are still widely used in underwater surveys. The main contribution of this paper is to design a level set formulation that is well suited to reconstruct fishes from acoustic images captured using multi-beam echo sounders. The evolution of the level set equation is coupled with a solution of another differential equation that effectively removes the noise, enabling the level set to converge to the objects of interest in the image.

Speckle noise in acoustic images is generally modeled by the Rayleigh distribution [44, 43]. Quidu et al. [108] estimate an optimal image filter size to compute an estimate of a good threshold by pixel correlation. Gagnon [49] shows numerical results of a wavelet domain based method for noise removal. Chen and Raheja [29, 15] show a wavelet lifting based method where the spatial correlation of acoustic speckle noise is broken by multiresolution analysis. In another approach to use wavelet based methods, Isar et al. [64] present a Bayesian-based algorithm. In a novel attempt to use the Markov Random Field (MRF) to segment acoustic images, Mignotte et al. [91] use an unsupervised scheme by employing an iterative method of estimation called Iterative Conditional Estimation (ICE). The authors used a maximum likelihood estimation to compute the MRF prior model.

Krissian et al. [73] provide a variation of the anisotropic diffusion process [106] constrained by speckle noise model. Anisotropic diffusion provides an intelligent way to perform diffusion without affecting prominent edges in an image.

Level set based methods have been shown to successfully restore noisy images [113]. Osher and Rudin [102] developed shock filters for image enhancements. Malladi and Sethian [87] have shown image smoothing and enhancement based on curvature flow interpretation of the geometric heat equation. In a more recent approach to use level set methods for acoustic image segmentation, Lianantonakis and Petillot [82] provide an acoustic image segmentation framework using the region based active contour model of Chan and Vese [25]. The authors comment that the level set based model has good regularization properties similar to those of a Markov random field [82].

A relevant work by Balabanian et al. [15] shows an interactive tool for visualization of acoustic volumetric data using a well known volume visualization technique called Ray-casting. Authors in [15] develop a tool for manual selection by region growing and visualization of moving fish schools using graphics hardware. The work presented in this paper is intended to develop mathematical models to automatically extract meaningful features from acoustic data with no user interaction. This work does not provide any tool for visual analysis, rather it presents a computational framework for noise suppression and 3D reconstruction.

This paper concentrates on using the level set methods for simultaneous suppression of noise and 3D reconstruction of relevant features. We limit features of interest to fishes from acoustic images and provide a level set based framework for acoustic image segmentation. Image restoration techniques based on level set evolution are generally oriented to segment the image or to remove noise from it. Work by Lianantonakis and Petillot [82] is closest to our approach since they use active contours using Mumford-Shah functional for seabed classification, but together with extraction of Haralick feature set for textural analysis. Our method differs from theirs since it is not possible to rely on texture based classification in the absence of any specific textures in the image.

Since acoustic data resulting from marine surveys can result in gigabytes of information, we employ GPU (Graphics Processing Unit) based computations for 3D reconstruction. The GPU is not very suitable for data intensive applications due to unavailability of large memory on commodity hardware. A number of publications suggest schemes to circumvent this situation by performing computations in a streaming manner [81, 79, 54], but most of the implementations process 2D sections to generate a 3D reconstruction. We present a Level Set method implementation with computations performed entirely in 3D using the 3D textures (read only) available to the CUDA 2.0 framework. CUDA (Compute Unified Device Architecture) is a parallel programming model and software environment designed to develop application software that transparently scales its parallelism to leverage the increasing number of processor cores on the GPU. It allows programming computationally intensive algorithms to take advantage of the available graphics hardware. Our method is streaming and is optimized for memory usage, consuming only twice the CPU memory of the input volume.

The paper is organized as follows. In section 5.2, we present the preliminaries on the active contour model. In section 5.3 we discuss the work of Chan and Vese [25] on minimizing the Mumford-Shah functional in images. Section 5.4 outlines our work on the noise suppression model which is solved together with the level set equation. Section 5.5 details our CUDA implementation for 3D reconstruction of the fishes based on the parallelization of the results of Section 5.4. In Section 5.6, we present the experimental results, and conclude the paper in section 5.7.

5.2 Background

Let an image $I(x, y)$ be defined on a bounded open subset $\Omega : \{(x, y)|0 \leq x, y \leq 1\}$ of $\mathbb{R}^2$, with $\partial\Omega$ as its boundary. I takes discrete values between 0 and $(2^n - 1)$ where n is the number of bits used to store intensity values. The basic idea in active contour model is to evolve a curve $C(s) : [0, 1] \mapsto \mathbb{R}^2$ by minimizing the following energy functional [101]:

$$E(C) = \alpha \int_0^1 |C'|^2 \, ds + \beta \int_0 |C''| \, ds - \lambda \int_0^1 |\nabla I(C)|^2 \, ds,$$

where α, β, and λ are positive parameters. In the above energy functional, the evolution of curve C is controlled by the internal energy (first two terms that define the smoothness of the curve) and the external energy (the last term that depends on the edges present in the image). The curve C can be represented by an implicit function ϕ, $C = \{(x, y)|\phi(x, y) = 0\}$, where the evolution of C is given by the zero level curve at any time t of the function $\phi(x, y, t)$.

With this formulation, an edge detector is defined as a positive decreasing function $g(\nabla I)$ based on the gradient of image [106] such that

$$\lim_{|\nabla I| \to \infty} g(\nabla I) = 0.$$

Therefore, the zero level curve evolves in the normal direction and stops at the desired boundary where g vanishes.

Evolving the curve C in normal direction amounts to solving the partial differential equation (PDE) [103]

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| F, \tag{5.1}$$

with the initial condition $\phi(x, y, 0) = \phi_0(x, y)$, where $\phi_0(x, y)$ is the initial contour. Motion by mean curvature allows for cusps, curvature and automatic topological changes [103, 26]. This results in the speed function $F = \text{div}\left(\frac{\nabla \phi}{\|\nabla \phi\|}\right)$ in terms of the curvature of ϕ

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| \text{div}\left(\frac{\nabla \phi}{|\nabla \phi|}\right), \text{ with } \phi(x, y, 0) = \phi_0(x, y),$$

where div$(\cdot)$ is the divergence operator, and $|\cdot|$ is the L_2 norm.

5.3 Minimizing the Mumford-Shah functional in image

Chan and Vese [26] provide an alternative approach to the edge based stopping criterion. The authors suggest the stopping term based on Mumford-Shah segmentation techniques [95]. The motivation behind using this alternative stopping term is that in many cases, the edges in an image are not very well defined. Either it is ambiguous to position the edges across the gradient due to smoothly varying intensities [26] or it is difficult to select prominent edges due to presence of noise (as in the case of acoustic images). The method of Chan and Vese [26] is minimization of an energy based segmentation. Assuming that the image I is composed of two regions of piecewise constant intensities of distinct values I^i and I^o, and that the object of interest is represented by I^i, we define the curve C to be its boundary. Using the Heaviside function H

$$H(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases},$$

and the Dirac-Delta function δ_0

$$\delta_0(z) = \frac{d}{dz} H(z),$$

the energy functional is formulated as

$$\begin{aligned} E(c_1, c_2, C, t) = & \mu \int_\Omega \delta_0(\phi(x,y,t)) |\nabla\phi(x,y,t)| \, dx \, dy \\ & + \nu \int_\Omega H(\phi(x,y,t)) \, dx \, dy \\ & + \lambda_1 \int_\Omega |I(x,y) - c_1|^2 \, dx \, dy \\ & + \lambda_2 \int_\Omega |I(x,y) - c_2|^2 \, dx \, dy, \end{aligned} \tag{5.2}$$

where $\mu \geq 0$, $\nu \geq 0$, $\lambda_1, \lambda_2 > 0$ are fixed parameters. c_1 and c_2 are average intensity values inside and outside C. The constants c_1 and c_2 can also be written in terms of I and ϕ as

$$c_1 = \frac{\int_\Omega I(x,y) H(\phi(x,y,t)) \, dx \, dy}{\int_\Omega H(\phi(x,y,t)) \, dx \, dy}, \tag{5.3}$$

$$c_2 = \frac{\int_\Omega I(x,y)(1 - H(\phi(x,y,t))) \, dx \, dy}{\int_\Omega (1 - H(\phi(x,y,t))) \, dx \, dy}. \tag{5.4}$$

The variational level set approach gives the following Euler-Lagrange equation [26]

$$\frac{\partial \phi}{\partial t} = \delta_\epsilon(\phi) \left[\mu \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} - \nu - \lambda_1 (I - c_1)^2 + \lambda_2 (I - c_2)^2 \right] \tag{5.5}$$

with the initial condition, $\phi(x,y,0) = \phi_0(x,y)$ and the regularized Dirac-Delta function δ_ϵ,

$$\delta_\epsilon(z) = \frac{\partial}{\partial z} H_\epsilon(z) = \pi^{-1} \epsilon^{-1} \left(1 + \frac{z^2}{\epsilon^2}\right)^{-1}, \tag{5.6}$$

where the regularized one-dimensional Heaviside function H_ϵ is given by

$$H_\epsilon(z) = \frac{1}{2} \left(1 + \frac{2}{\pi} \tan^{-1}\left(\frac{z}{\epsilon}\right)\right).$$

Despite the fact that this model has advantages over the edge based model in that it is able to detect boundaries with smoothly varying intensities and blurred edges, the main limitation comes from the fact that it can only discriminate regions with different mean intensities [82]. In particular, strong textures pose a problem with this approach. Lianantonakis and Petillot [82] solve this problem by extracting the Harlick feature set based on the co-occurrence matrix.

The acoustic images considered by Lianantonakis and Petillot [82] are of the seabed. Such images show strong textural variations of the bottom surface of the sea. In this paper, we restrict ourselves to acoustic images of freely swimming fishes. While such images are also corrupted by speckle noise, they do not show specific textural patterns. Figure 5.1(a) shows part of such an image where the fish cross sections are discriminated by very high intensities compared to the background. The presence of reflectance from air bubbles mixing into water, also contribute to the noise. While working with level sets, a standard procedure is to keep ϕ to a signed distance function [101]. A direct application of the level set equation given by equation (5.5), with $\phi(x, y, 0) = 0$ initialized to set of squares regularly distributed over the image, shows that the evolution of the level set eventually stops at the wrong place (see Figure 5.1(b)). Furthermore, lack of any specific textural patterns leads us to formulate a successive noise suppression scheme where the Mumford-Shah energy functional is minimized while simultaneously removing noise from the image. The later aids in fast convergence of the level curve in our formulation.

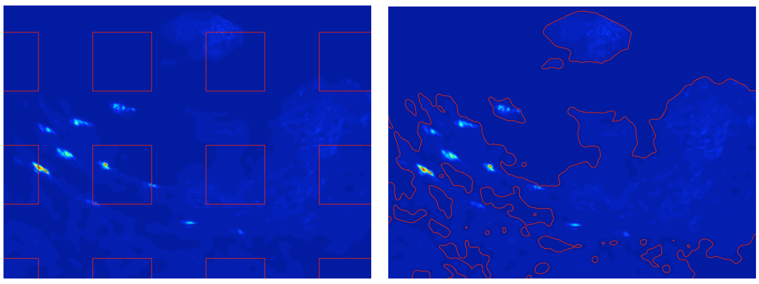

(a) Initialization contour. (b) Result at convergence.

Figure 5.1: *Application of the level set equation (5.5).*

5.4 Noise suppression model

As discussed before, acoustic images suffer from heavy speckle noise. At first thought, it might sound reasonable to apply a global threshold to the image to get rid of the noise. However, this is not a plausible option since for a particular chosen threshold there might be echo intensities of fishes lower than it and therefore such a threshold will result in loss of information [126, sec. 5.4.6, 6.3]. An adaptive threshold might also not provide a solution since the speckle has a high local intensity, and therefore could show false positives. Therefore, we resort to the global energy minimizing method to suppress noise.

Considering the image I to be time varying, the basic idea behind noise suppression is to solve the noise suppression equation as an update step to the level set equation resolution in a single pass. The noise suppression equation is

$$\frac{\partial I(x,y,t)}{\partial t} = k \cdot \max(0, \hat{c} - I(x,y,t)), \tag{5.7}$$

where k is a constant and $\hat{c}$ is a scalar parameter that is computed as an optimal threshold at any time step t based on $\phi(x,y,t)$.

The computation of $\hat{c}$ is based on the bounded subset I^i given by

$$I^i(x,y,t) = I(x,y,t) \cdot H_\epsilon(\phi(x,y,t)).$$

The values given by the set I^i are used to compute the weighted median [145] as shown in algorithm 5.1 which is used as $\hat{c}$ at that particular time step t.

```
**Input**: $I(x,y,t)$, $H_\epsilon(x,y,t)$
**Output**: $\hat{c}$
$V = \{v_i : v_i = I(x,y,t),\ x \in [1,l],\ y \in [1,m],$
                    $i \in [1,n],\ n = l \cdot m\}$
$W = \{w_i : w_i = H_\epsilon(x,y,t),\ x \in [1,l],\ y \in [1,m],$
                    $i \in [1,n],\ n = l \cdot m\}$
Sort $V$ in ascending order
$W \leftarrow W \backslash \{w_z\}\ \forall w_z = 0$
$V \leftarrow V \backslash \{v_z\}, \{v_z : v_z \in V,\ \forall z \text{ where } w_z = 0\}$
$S \leftarrow \sum_{k=1}^{n} w_k,\ w_k \in W$
Find index $i$ such that $\sum_{k=1}^{i} w_k \leq \frac{S}{2}, w_k \in W$
Find index $j$ such that $\sum_{k=j}^{n} w_k \leq \frac{S}{2}, w_k \in W$
Median $M = \{v_i, v_j\}$
$\hat{c} \leftarrow \min(v_i, v_j)$
```

Algorithm 5.1: Computation of weighted median.

The use of median filtering to remove noise is not new in image processing [52, 85]. We now show that the estimate of $\hat{c}$ based on the weighted median is a good approximation for the gray-level threshold that separates the noise from the signal, and is robust in a way that the evolution of the level set converges with increasing t.

$H_\epsilon(z)$ attains values close to zero for regions outside C and values close to one inside C. In fact, $\lim_{z \to \infty} H_\epsilon(z) = 1.0$ and $\lim_{z \to -\infty} H_\epsilon(z) = 0.0$. At the start of level set evolution, I^i covers most of Ω and therefore, $H_\epsilon(z)$ attains values close to one for most of the intensity values. This results in computation of $\hat{c}$ which is equivalent to an unweighted median for values in I^i. A median is the central point which minimizes the average of absolute deviations. Therefore, a median better represents the noise level when the data contains high intensity values that are fewer in number, and a majority of intensity values that correspond to the noise. As a result, the initial iterations of the solution suppress the intensity values that are less than the median to a constant level (the median itself). One should expect the median value to increase as the level set contracts, but since we use a regularized Heaviside function as weight for the intensity values, the weighted median converges to zero since most of I contains intensity values of zero with near-zero weight.

Other variations of estimation of $\hat{c}$ are certainly possible, but we find that a weighted median based approach results in effective noise removal with very little information loss. For instance, a value of $\hat{c}$ taken to be c_1, the mean intensity inside C, does a similar suppression but with a high signal loss compared to the former. Furthermore, the mean does not converge as fast as the median does and might result in relatively higher values for large fish cross sections. It must be noted however, that the computation of the median is costly as compared to that of the mean.

5.5 CUDA implementation for 3D reconstruction

Equation (5.5) can be solved by discretization and linearization in ϕ[26]. Discretization of equation (5.7) in I gives

$$\begin{aligned}\frac{I_{n+1}(x,y)-I_n(x,y)}{\Delta t} =& k\cdot\max\left(0,\hat{c}-I_n(x,y)\right)\\ =& \left\{\begin{array}{ll} 0, & \text{if } I_n(x,y)\geq\hat{c}\\ k\cdot(\hat{c}-I_n(x,y)), & \text{otherwise}\end{array}\right., \end{aligned} \tag{5.8}$$

with $k=\frac{1}{\Delta t}$, and $t_{n+1}=t_n+\Delta t$. The above time discretization yields

$$I_{n+1}(x,y)=\left\{\begin{array}{ll} 0, & \text{if } I_n(x,y)\geq\hat{c}\\ \hat{c}-I_n(x,y), & \text{if } I_n(x,y)<\hat{c}\end{array}\right. . \tag{5.9}$$

Acoustic images captured by echo-sounders are generally taken as planar image scans by moving the echo-sounder in one direction, thereby sweeping a volume. Let us denote individual images as $I(x,y,\tau)$ for images taken after every $\delta\tau$

time interval. A volume is constructed by stacking these individual images in sequence and applying geometric correction for distance $\delta\tau(v)$ between individual slices, where v is the instantaneous speed of the instrument (the current data was captured with constant unidirectional instrument velocity). It must also be noted that the individual acoustic images are obtained from a set of acoustic intensity signals along beams by a polar transformation. The level set equations for curve evolution in $\mathbb{R}^2$ extend uniformly to surface evolution in $\mathbb{R}^3$. The second differential equation also holds true for noise suppression in a volume. Therefore, it is possible to reconstruct 3D moving fishes with the level set evolution of these equations combined.

Processing a huge dataset demands that a minimum of memory is consumed. We propose to keep two volumes in the host memory, one for the intensity values (I) and the other for the signed distance function (the implicit function, ϕ). The CPU manages the memory scheduling by dividing the volumes into small subvolumes that can be processed on the GPU. We keep two small 3D textures of size $128 \times 128 \times 128$, I_{GPU} and ϕ_{GPU}. A complete level set update is divided into a set of subvolume updates. Each subvolume in the two volumes is fetched to the GPU via 3D textures (read only, but with good cache coherence). Results of computations are written to CUDA memory and then transferred back to the CPU volumes. A simplified diagram of this is shown in Figure 5.2.

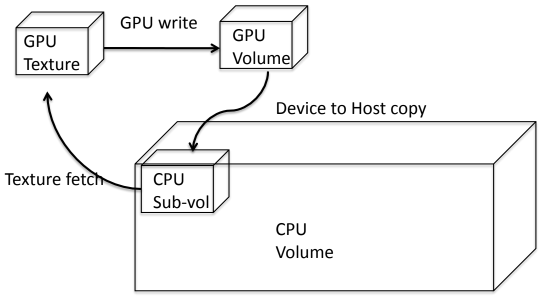


Figure 5.2: *Streaming computation.*

CUDA exposes a set of very fast 16 KB shared memory available to every multiprocessor in a GPU. However, a 16 KB memory chunk is shared only between a thread block, and thus to make use of it the application must load different data for different blocks. Furthermore, the 16 KB limit poses a restriction on the amount of data that can be loaded at any point of time. Here, we use 3D textures for reading the data. Since we do not want to write back to the same texture (before a single step of filtering is complete), using the read-only 3D textures available to CUDA is a natural choice. 3D texturing has hardware support for 3D cache which accelerates any texture reads in succession. To load a 3D data (a small subset of the volume) from the global memory into the shared memory could be a little tricky and might not result in the same performance as provided by the specialized hardware for 3D texture cache. In our application, data writes are made to the global memory. The latency in writing is hidden in the data processing since we do not synchronize the threads until the end of a subvolume processing. These can further be optimized by making use of coalesced writes.

The solution of the PDE is computed in iterations over the full volumes. Following are the CUDA kernels that were used in the updates.

5.5.1 Signed distance transform

Signed distance transform is a global operation and cannot be implemented in a straightforward manner. We compute a local approximation of the Euclidean distance transform using the Chamfer distance. A narrow band distance transform is computed layer by layer using, what we call a *d*-pass algorithm. Every pass of the method adds a layer of distance values on the existing distance transform. The distance values are local distance increments computed in a $3 \times 3 \times 3$ neighborhood. Therefore, every single pass needs only local information to compute the distance values except at the border of the sub-volume. We therefore support every sub-volume with a one voxel cover from other adjoining sub-volumes, thereby reducing the computational domain to a volume of size $126 \times 126 \times 126$. The CPU scheduler takes care of the voxel cover. At the beginning, the interface (zero level) is initialized to a used specified bounding cuboid or a super-ellipsoid.

5.5.2 Average intensities

Computing average intensities (c_1 and c_2) is an operation that cannot be easily computed in a parallel fashion, and a reduction like method is required for the same. We employ a slightly different scheme to compute averages by using three accumulator sub-volumes on the GPU. These accumulators are essentially 3D sub-volumes of the same dimensions as of the textures. Every voxel in the accumulators accumulates (adds up), the values for H, $I \cdot H$, and $I \cdot (1 - H)$ for all the sub-volumes in the CPU volume(s). We then sum up the small sub-volume on the CPU to get the final sum and compute c_1 and c_2 values from it. Using a mixed mode CPU-GPU computation not only reduces the complexity of an inherently non-parallel operation, but also performs better by moving less expansive parts of the computation to the CPU.

5.5.3 Median computation

Computing median on the GPU is not very straightforward since it is an order statistic and requires that the data be sorted. Therefore the computation of weighted median is very different than the one for average intensity value. Since sorting values of order of millions in every iteration of the solver is not a computationally good solution, we resort to the alternative definition of the median. A median is a value that divides the data-set into two sets of equal cardinalities. This definition is generalized for a weighted median. Therefore, for a data-set V with weights W associated with each value in the set, the median value V_k is the value for which

$$\sum_{i=0}^{k} W_i = \sum_{i=k+1}^{n} W_i.$$

This equation can only be solved iteratively, starting with a guess index value k_0. In our CUDA implementation, we start with V_{k_0} to be the mean value c_1 and iteratively reach the weighted median. In every iteration, the increment $\triangle i$ for the index k_0 is computed as

$$\triangle i = \begin{cases} \dfrac{\displaystyle\sum_{i=0}^{k} W_i - \sum_{i=k+1}^{n} W_i}{\displaystyle\sum_{i=0}^{k} W_i}, & \text{if } \sum_{i=0}^{k} W_i > \sum_{i=k+1}^{n} W_i \\ \dfrac{\displaystyle\sum_{i=k+1}^{n} W_i - \sum_{i=0}^{k} W_i}{\displaystyle\sum_{i=k+1}^{n} W_i}, & \text{if } \sum_{i=k+1}^{n} W_i > \sum_{i=0}^{k} W_i \end{cases} .$$

The increment $\triangle i$ can be adaptively controlled to give results as precise as desired.

5.5.4 Solver update

A PDE update in the level set method comprises of computing the curvature energy and the external energy. In order to compute the curvature term (involving double derivatives) for a voxel in a sub-volume by centered differencing, we need information from a $5 \times 5 \times 5$ neighborhood with the current voxel at its center. Therefore, the sub-volume size needs a cover of two voxels on all sides, thus reducing the computational domain further down to $124 \times 124 \times 124$. The memory scheduler performs additional computations to effectively cover the whole volume with the new setup. Once the energy terms are computed, the PDE solver kernel updates ϕ_{GPU} and uses c_1 to update I_{GPU}. These sub-volumes are then updated to the CPU main volume.

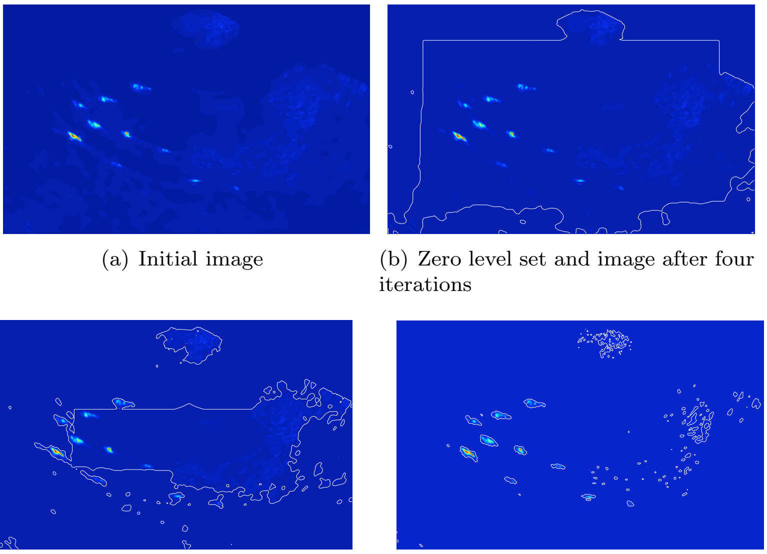
It is often convenient to perform anisotropic diffusion on the input image so that the evolution of the level curve is smooth and ϕ is well behaved. Finally, the zero level surface is extracted from the evolved ϕ using the Marching-cubes method.

5.6 Experimental results

We present experimental results on sample acoustic 2D images to show that the suppression scheme works well on such images. Figure 5.3 shows evolution of the level set. The parameters for this evolution were chosen to be: $\mu = 0.0005$, $\nu = 0$, $\lambda_1 = \lambda_2 = 1$, and $\epsilon = 2.5$. It can be seen that the original image suffers from speckle noise as seen in Figure 5.4 and that the final zero level contour approximates the fish boundaries very well.

We next show results of application of the level set equation and the noise suppression scheme on a small 3D volume of size $150 \times 100 \times 50$. Fish intensities can be identified in dark green against a noisy background. The level set equation was initialized with the zero level set of ϕ_0 as the bounding box of the volume. The level set is then allowed to evolve with parameters, $\mu = 0.0005$, $\nu = 0$, $\lambda_1 = \lambda_2 = 1$, and $\epsilon = 1.0$. Figure 5.5 shows the evolution at different time steps and the final level surface.

We test the CUDA solver on a larger volume of size $686 \times 1234 \times 100$. This volume uses about 470 MB of CPU memory along with the same amount of memory consumed by the signed distance field. Figure 5.6 shows the extracted fish trails. We test our implementation with the mobile GPU, GeForce 8600M GT (Nvidia CUDA compute capability of 1.1) with 256 MB of memory on a Mac OS X notebook with 2 GB of host memory. The total number of iterations required until convergence were 29, with a compute time of about 52 seconds per iteration (32 seconds without median computation). The signed distance field was reconstructed in a narrow band of width 20 voxels in every iteration. With the commodity graphics hardware, we expect to get better speedups. Fur-


(a) Initial image

(b) Zero level set and image after four iterations

(c) Zero level set and image after ten iterations

(d) End of evolution after 16 iterations

Figure 5.3: *Level set evolution on sample image.* $\epsilon = 2.5$.

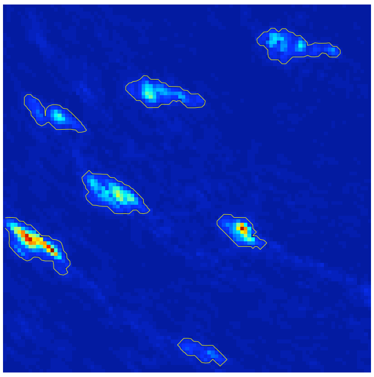
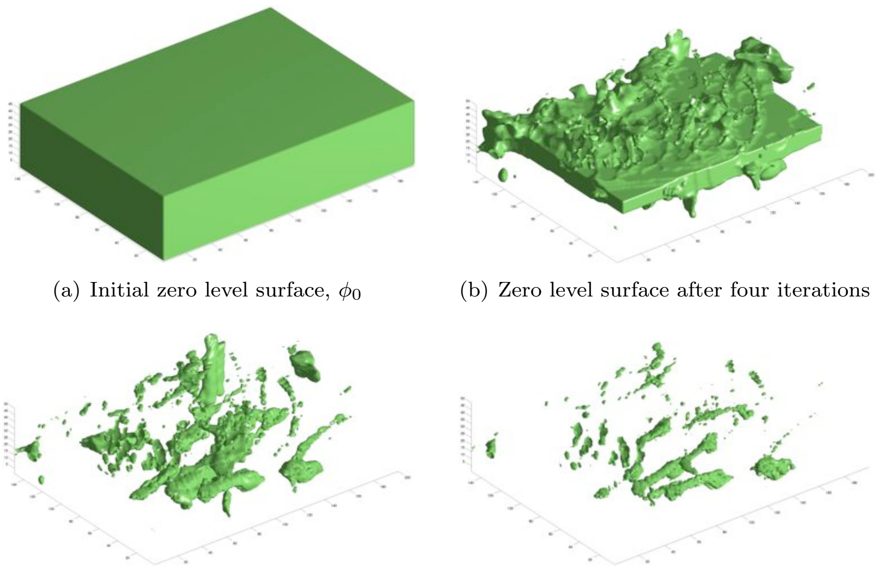


Figure 5.4: *The final contour shown on the part of the original image.*

thermore, a better GPU with more onboard memory should allow loading larger subvolumes, thus reducing the overhead of multiple memory transfers.

In order to compare the 2D and 3D reconstructions, we show an overlay of 2D



(a) Initial zero level surface, ϕ_0

(b) Zero level surface after four iterations

(c) Zero level surface after six iterations

(d) End of evolution after nine iterations

Figure 5.5: *Level set evolution on sample volume.* $\epsilon = 1.0$.

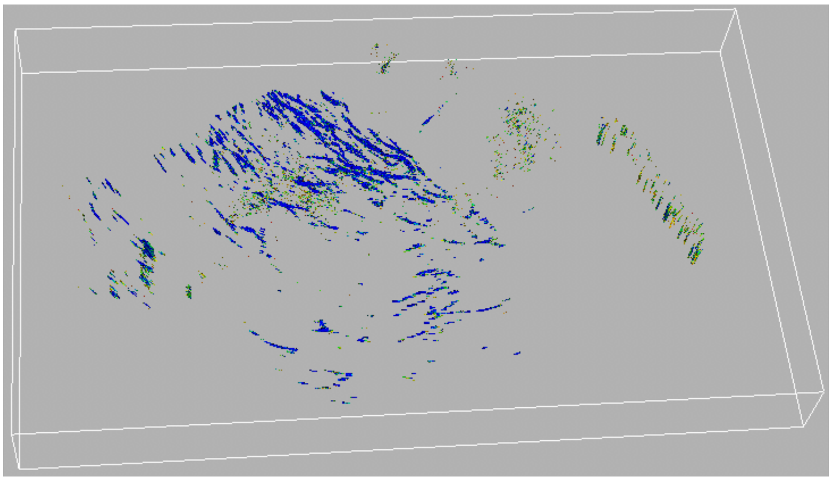


Figure 5.6: *Fishes extracted from a volume of size* $686 \times 1234 \times 100$.

curves over the extracted 3D surface. This is shown in Figure 5.7. The results agree very well when the 2D image contains high intensity objects. The acoustic images were taken by scanning fishes in an aquarium and the images corresponding to the bottom of the aquarium (time slices with higher depth, 30 to 50 in Figure 5.7) contain almost no fishes. Therefore, these images contain very little useful information. The 2D level set evolution fails to detect fishes in these images. It is also worth mentioning that the suppression of noise is based on weighted median and if the images do not contain high intensities, it is possible that the estimated threshold value does not accurately represent the noise level. Therefore, the 3D results should be trusted since the 2D reconstruction does not consider information present in other image planes. We would like to comment that a ground truth segmentation is not practically possible for open sea. Evaluation of the extracted fish trails/schools by domain experts is under process because of marine surveys.

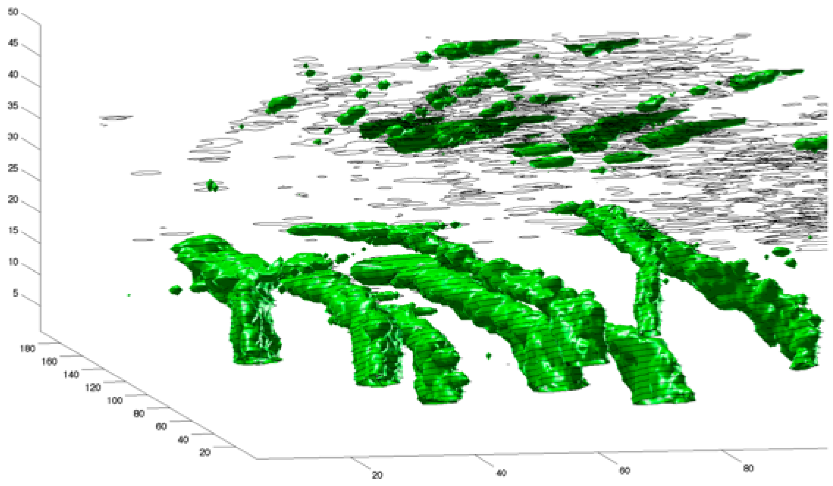


Figure 5.7: *Comparison of zero level 2D curves with the zero level 3D surface.*

While we claim that this method works on acoustic images with high variance in intensity values resulting in a binary segmentation of the image, it is certainly possible to perform a segmentation resulting in more than two segments [25].

5.7 Conclusions

In this paper, we presented augmentation of level set formulation based on the Mumford-Shah functional to a noise suppression scheme, well suited for object reconstruction from acoustic images. Our method is based on computation of a threshold by weighted median of intensity values. We prove that the method converges with evolving level set and show that the experimental results comply with that. We show a 3D reconstruction of objects from time series images which is useful in tracking moving objects and to observe their kinetics. An optimized GPU based implementation has been presented for streaming computation of the large volumetric data.

Acknowledgement

The acoustic data was collected in collaboration with the Danish Institute of Fisheries Research (DIFRES) in Denmark, and the authors are thankful for their support.

Chapter 6

Fast streaming 3D levelset segmentation on the GPU for smooth multi-phase segmentation

O. Sharma[†], Q. Zhang[‡], F. Anton[†], and C. Bajaj[‡]

[†]Department of Informatics and Mathematical Modelling,
Technical University of Denmark, Lyngby, 2800, Denmark

[‡]Computational Visualization Center
The University of Texas at Austin, Austin, Texas, 78712-0027, USA

Abstract

Level set method based segmentation provides an efficient tool for topological and geometrical shape handling, but it is slow due to high computational burden. In this work, we provide a framework for streaming computations on large volumetric images on the GPU. A streaming computational model allows processing large amounts of data with small memory footprint. Efficient transfer of data to and from the graphics hardware is performed via a memory manager. We show volumetric segmentation using a higher order, multi-phase level set method with speedups of the order of 5 times.

6.1 Introduction

Volume segmentation is a computationally demanding task. We address this problem by employing a fast solution to the involved partial differential equations (PDEs) using the graphics processing unit (GPU). A recent trend in solving computationally expensive problems is to redesign the solution of the problem so that it can take advantage of the high arithmetic parallelization capability of the GPUs. We propose a novel GPU based framework for level set segmentation of large volumes.

A segmentation problem is to subdivide a three dimensional image $I(x, y, z) : \Omega \mapsto \mathbb{R}$, where Ω is a bounded open subset of $\mathbb{R}^3$, into non-overlapping partitions $\Omega_i (i = 1..\, n_c)$ such that $\bigcup \Omega_i = \Omega$, where each partition is homogeneous in the sense that it minimizes a certain quantity. Each region is said to produce a class representing the partition.

Implicit surfaces naturally capture the topology of the underlying surface in contrast to explicit or parametrized surfaces. Therefore level set based methods are very useful in this context. Deformable level set surfaces under mean curvature flow provide an intuitive means of segmentation. The pioneering work by Osher and Sethian [101] presents an effective implicit representation for evolving curves. Later on, the work was developed in context of the Mumford-Shah functional by Chan and Vese [27] for 2D images that do not contain prominent edges. In a more advanced paper, these authors suggest a multi-domain segmentation [137] using the same level set framework. Other variants of the same method exist for applications like image de-noising based on total variation minimization [111]. Conventional level set methods solve the interface evolution equation with linear interpolation of the implicit function and its derivatives (at sampled grid points). The resulting level set surface is C^0. Bajaj et al. [10, 13] present a cubic spline based level set method that produces a C^2 level set surface.

Due to the high computational intensity of the level set method and inherent parallelism in the solution of the involved PDEs, a parallel compute environment is best suited. Schemes for fast evaluation of PDEs are suggested by Weickert et al. [143]. Multigrid methods are also suitable for a fast solution of differential equations. An active contour model using multigrid methods is suggested by Papandreou and Maragos [104]. Of particular interest is a solution to the level set equations for segmenting large volumes. GPU based implementations have been proposed in [133, 80], among which Lefohn et al. [80] demonstrate an efficient sparse GPU segmentation using level set methods.

In this work, we propose a streaming solver framework suited for large volume segmentation. With 3D textures available to the commodity graphics hardware, we show that a 2D slicing is no longer required for a solution. This is also in contrast to [80], where the authors use a compact representation of the active volume packed into 2D textures. We solve the governing partial differential equations (PDEs) for a general case of any number of segmentation classes. The number of classes is determined as the level set evolves, creating new classes while merging some of the existing ones. Every single class then gives rise to a partition of the volume. The result of the streaming solver is demonstrated with multi-domain segmentation along with speedup benchmarks for tri-linear and tri-cubic level set computations.

6.2 Level set segmentation

The main idea behind a level set based segmentation method is to minimize an energy term over an open domain by numerically solving the corresponding time varying form of the variational equation. Let us represent a volume by a scalar field $I(x, y, z) : \Omega \mapsto K$, where Ω is a bounded open subset of $\mathbb{R}^3$, and $K \subset \mathbb{R}$ is a bounded set of discrete intensity values sampled over a regular grid. In this setup, motion by mean curvature provides a deformable level set formulation where the surface of interest moves in the direction of the normal at any point with velocity proportional to the curvature [101].

The deformable surface is represented by a level set of an implicit function $\phi(x, y, z) : \Omega \mapsto \mathbb{R}$. In level set methods, ϕ is generally chosen to be a signed distance function since it allows mean curvature flow with unit speed normal to the level set interface [101, chap. 6]. Toward a segmentation approach, various energy formulations are possible. The energy functional is further penalized by a regularizing term that introduces smoothness in the resulting surface. The Mumford-Shah energy functional has a distinct advantage of producing better segmentation regions in absence of sharp edges as compared to an edge based energy functional. Consider an evolving interface $\Gamma = \{(x, y, z) : \phi(x, y, z) = 0\}$ in Ω, denoting $\Gamma^+ = \{(x, y, z) : \phi > 0\}$ as the interior of the volume bounded by Γ and $\Gamma^- = \{(x, y, z) : \phi < 0\}$ as the exterior of the volume bounded by Γ.

A modified Mumford-Shah energy can be written as:

$$\begin{aligned} F(c_1, c_2, \Gamma) &= \mu \cdot Area(\Gamma) + \nu \cdot Volume(\Gamma^+)) \\ &\quad + \lambda_1 \int_{\Gamma^+} |I - c_1|^2 \, dx\, dy\, dz \\ &\quad + \lambda_2 \int_{\Gamma^-} |I - c_2|^2 \, dx\, dy\, dz, \end{aligned} \tag{6.1}$$

where $\mu \in \mathbb{R}_{\geq 0}, \mu \in \mathbb{R}_{\geq 0}, \lambda_1 \in \mathbb{R}_{>0}$, and $\lambda_2 \in \mathbb{R}_{>0}$ are fixed control parameters. In order to derive a variational form of (6.1), the energy F is regularized and minimized. Minimization by application of the Green's theorem and variational calculus yields the following time varying form of (6.1) [27]:

$$\frac{\partial \phi}{\partial t} = \delta_\epsilon(\phi) \left[\mu \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right) - \nu - \lambda_1 (I_0 - c_1)^2 + \lambda_2 (I_0 - c_2)^2 \right], \tag{6.2}$$

where, $\delta_\epsilon(\phi)$ is a regularized Delta-Dirac function [27], c_1 and c_2 are averages in Γ^+ and Γ^- respectively. The level set PDE (6.2) represents the mean curvature flow with a Mumford-Shah like image energy. Bajaj et al. [10] propose to solve the higher order regularizing term in (6.2) by cubic spline interpolation to compute accurate higher order derivatives of ϕ, thus avoiding numerical differentiation, which is very unstable.

6.3 Multi-phase, higher-order level set method

Equation (6.2) defines two decompositions of Ω with respect to the zero level set surface of ϕ, i.e., $\phi > 0$ and $\phi < 0$. Often in segmentation, we need more than two partitions of the input signal. Vese and Chan [137] show that multiple level set evolutions can be used to keep track of multiple regions in the signal. In a Multi-domain setup a single implicit function ϕ is replaced by a vector valued $\mathbf{\Phi} = \{\phi_0, \phi_1, \ldots, \phi_{m-1}\}$ function where m is the total number of implicit functions that are combined to give a maximum of $n = 2^m$ partitions of Ω. Equation (6.2) is replaced by a system of m PDEs. We compactly write this system as:

$$\begin{aligned} \frac{\partial \phi_i}{\partial t} = \delta_\epsilon(\phi_i) \Bigg[\mu \nabla \cdot \left(\frac{\nabla \phi_i}{|\nabla \phi_i|} \right) - \nu - \sum_{k=0}^{2^{m-1}-1} \Bigg\{ \big((I - c^1_{i,k})^2 \\ - (I - c^0_{i,k})^2 \big) \prod_{p=0}^{m-1,\, p \neq i} \left(b_{q,p} + (-1)^{b_{q,p}} H_\epsilon(\phi_p) \right) \Bigg\} \Bigg], \end{aligned} \tag{6.3}$$

for $i \in [0, m-1]$, where $H_\epsilon(z) = \frac{1}{2}\left(1 + \frac{2}{\pi}\tan^{-1}\left(\frac{z}{\epsilon}\right)\right)$ is a smooth version of the Heaviside function and

$$c^0_{i,k} = \text{mean}(I) \text{ in } (x,y,z) : \begin{cases} \phi_p > 0, & \text{if } b_{q,p} = 1, \\ \phi_p < 0, & \text{if } b_{q,p} = 0 \end{cases}$$

$$\text{with } q = 2k - k \bmod 2^i, \forall p \in [0, m-1], \text{ and} \tag{6.4}$$

$$c^1_{i,k} = \text{mean}(I) \text{ in } (x,y,z) : \begin{cases} \phi_p > 0, & \text{if } b_{q,p} = 1, \\ \phi_p < 0, & \text{if } b_{q,p} = 0 \end{cases}$$

$$\text{with } q = 2k + 2^i - k \bmod 2^i, \forall p \in [0, m-1]. \tag{6.5}$$

Here, $b_{q,p} =$ p^{th} bit ($\in \{0,1\}$) in the binary representation of q (either from (6.4) or (6.5)).

Consider $q \in \mathbb{Z}^+$ such that its i^{th} bit is 1 for Γ_i^+, 0 for Γ_i^-, and 0 if $i > (m-1)$. In this way, q spans the n possible regions induced by $\mathbf{\Phi}$. In (6.3), $c^1_{i,k}$ (or $c^0_{i,k}$) represents the average of intensity values of I in Γ_i^+ (or Γ_i^-) where the other bits determine regions inside or outside for rest of the implicit surfaces. To generate an index for a region, we enumerate the possible 2^{m-1} values and insert a 1 or a 0 at the i^{th} bit. Alternative expressions for q in (6.4) and (6.5) are

$$q = (k - (k \wedge (2^i - 1))) {<<} 1 + k \wedge (2^i - 1), \text{and} \tag{6.6}$$

$$q = (k - (k \wedge (2^i - 1))) {<<} 1 + 2^i + k \wedge (2^i - 1) \tag{6.7}$$

respectively, where $<<$ is the usual right-bitshift operator and $\wedge$ is the logical And operator. We find that the binary indexing for keeping track of various regions of $\mathbf{\Phi}$ is not only compact in the data structure sense (not in the topological sense), but also efficient for implementation on a constrained compute environment like the GPU.

6.4 Streaming solver architecture

In this section the streaming architecture of our GPU level set solver is presented. The solver operates on three-dimensional volumes and therefore, makes extensive use of 3D textures available to current GPUs. The complexity of the solver is increased by the fact that m simultaneous PDEs need to be solved to arrive at a solution to (6.3). The solver is based on the Nvidia CUDA compute framework [97]. The solver has a very small memory footprint on the GPU compared to the actual volume that it can handle. On the CPU side, memory requirement is of the order of the number of implicit functions. Our GPU computational setup consists of a host memory manager to handle data streaming and a set of CUDA kernels to operate on parts of the data fetched to the device by the host.

6.4.1 Compute Unified Device Architecture (CUDA)

Volume processing is a computationally intensive task since every voxel needs to be updated. Further, solving a differential equation on a regular grid by a numerical update method demands for even higher computational resources both in terms of clock cycles and memory. Today's graphics processing units (GPU) outperform any CPU in terms of raw computing power by factors of 10 and more. Nvidia provides a general purpose GPU computing framework known as the Compute Unified Device Architecture (CUDA) [97].

A GPU (also called a *device*) is a set of Streaming Multiprocessors (SMs) employing a new SIMT (Single-Instruction Multiple-Threads) multi-processor computational architecture. Each SM consists of eight Scalar Processor (SP) cores. In the CUDA framework, the threads are logically grouped into blocks and blocks are arranged in a grid. A device, acting as a coprocessor to the CPU (also called a *host*), is capable of running tens of thousands of threads at once (512 threads per block, with a maximum of 65535 blocks). Every SP has its own registers while every SM has access to small (16 KB) but high speed *shared memory*. The whole device has access to *device memory*, *constant memory*, and *texture memory*. The constant and texture memory are cached while the device memory is not. In fact, the shared memory can be viewed as a user managed cache.

The CUDA programming model exposes three basic concepts: a hierarchy of thread groups, shared memories, and barrier synchronization. C for CUDA is an extension to C with a minimal set of additions. A CUDA computation on the device takes place in the form of functions called *kernels* executed by every thread.

6.4.2 Memory manager

The solver is designed keeping in mind large volumes of data and therefore we stick to the streaming paradigm for processing. Central to our GPU computational setup is a *memory manager* that handles data streaming between the host and the device. Figure 6.1 shows a schematic diagram of the memory manager, which is responsible for the following:

- maintaining a memory hierarchy, and
- managing memory transfers between the host and the device.

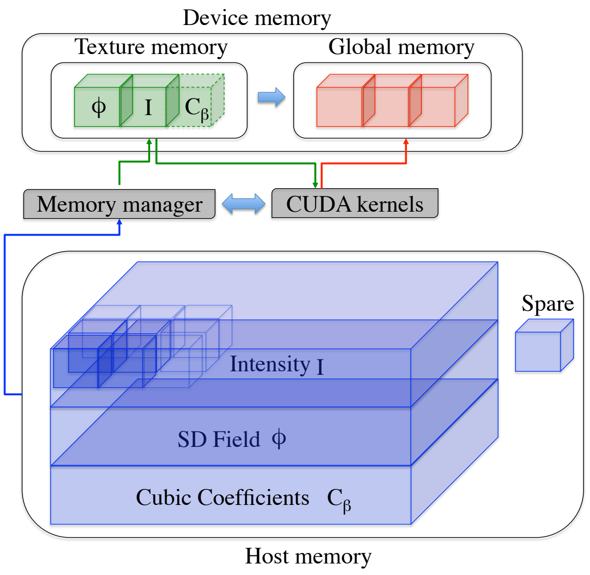


Figure 6.1: *The memory manager.*

Memory hierarchy

The data on the device is handled in manageable chunks of a 3D sub-volume called a *computational-volume*. The memory manager splits the entire volume into the minimum possible number of sub-volumes of size of the computational-volume. The size of such a computational-volume is chosen such that:

- computations are performed by one thread per voxel.
- the computational-volume fits into the device memory.

The computational-volume is further divided into the CUDA grid and block for thread invocation. CUDA allows for a 3D block of threads, but not a 3D grid of blocks. The memory manager builds a logical 3D grid that is mapped to a 1D grid of blocks, assigning each voxel with a thread to process it. For a $128 \times 128 \times 128$ computational-volume, a typical grid size could be $16 \times 16 \times 16$ blocks with each block consisting $8 \times 8 \times 8$ threads. Note that the number of threads in a block cannot exceed 512 with the version 2.0 of CUDA. The hierarchy of thread blocks, grid and the computational-volume is shown in Figure 6.2.

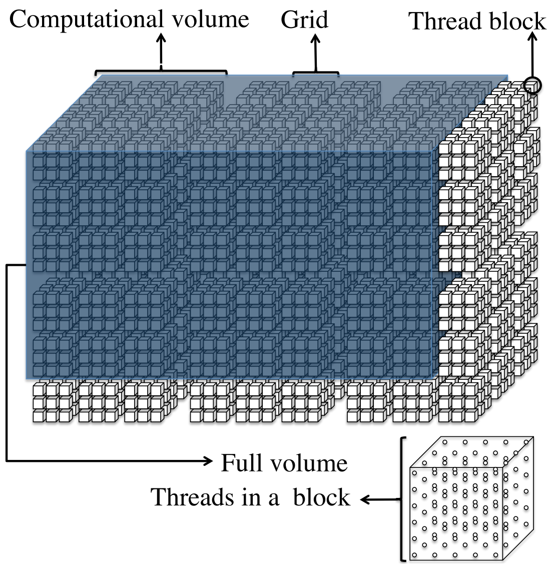


Figure 6.2: *Volume hierarchy.*

Memory transfers

For operations that involve accessing voxel neighbors (e.g., finite differencing or convolution filtering), the memory manager appropriately pads the computational-volume to enable the required number of shared voxels around the border of the computational-volume. This effectively reduces the size of the volume to incorporate neighbors along the border of the volume. Further, the full volume might not be an exact multiple of the computational-volume, therefore the memory manager pads the computational-volume on the boundary with null values for regions falling outside the main volume.

Memory copies between device and host are performed in size of the computational-volume. Special care is taken while copying data along the border of the full volume. The memory manager also dynamically allocates and frees the device memory if required by a kernel. Computational-volumes transferred to the device exist as 3D textures. Individual kernels then operate on these sub-volumes and the results are stored on the device's global memory, and are subsequently transferred back to the host volume(s).

6.4.3 Solver kernels

For solving the level set equation, the solver performs a series of operations. In the order of execution, these are:

1. Interface initialization,
2. Signed distance field computation,
3. Average values computation
4. Cubic coefficients computation, and
5. PDE time stepping.

To achieve maximum performance, we use hybrid CPU-GPU computations. Kernel specific details of the solver are explained next for the above mentioned operations. The majority of these steps can be executed in a streaming fashion with an exception of average value computation.

Interface initialization

The level set interface Γ is initialized to a bounding box or to a super-ellipsoid with center (c_x, c_y, c_z) and radii (r_x, r_y, r_z) of implicit equation

$$\left(\frac{x-c_x}{r_x}\right)^n + \left(\frac{y-c_y}{r_y}\right)^n + \left(\frac{z-c_z}{r_z}\right)^n = 1,$$

for a two domain segmentation. Multi-domain initialization should ensure that all the possible classes occupy non-zero regions in space at the start so that all the domains have scope of evolution. The domain Ω is decomposed into smaller sub-domains and each sub-domain is assigned small super-ellipsoids (with randomized centers) that form the interface Γ for each implicit function. Vese and Chan [137] observe a better and faster convergence in a 2D case for such an initialization. Our tests confirm this observation for the 3D case. Volumes for which specific geometry of the interested features to segment is known apriori, a different initialization can lead to better results with faster convergence.

The kernel module for interface initialization computes the implicit function $\mathbf{\Phi}$ such that:

$$\phi_i(x,y,z) = \begin{cases} k, & \text{if}\,(x,y,z) \in \Gamma^+ \\ -k, & \text{otherwise,} \end{cases}$$

where $i \in [0, m-1]$, and $k \in \mathbb{R}^+$ is a constant.

Signed distance field

The solver adopts a narrow band approach to constructing a signed distance field on the device using a d-pass approach to compute the distance field in d layers where $2d$ is the integer width of the narrow band. This is a streaming algorithm to create a Chamfer distance field [21] that uses optimal values of coefficients for distance multipliers to minimize accuracy error compared to an actual distance field (see Algorithm 6.1) [115]. The algorithm identifies the voxels belonging to the zero level set of ϕ and incrementally adds distance layers in Γ^+ (+ve distance layer) and in Γ^- (-ve distance layer). The order in which the 6 face-neighbors ($\mathcal{N}_6$), 12 edge-neighbors ($\mathcal{N}_{12}$) and 8 vertex-neighbors ($\mathcal{N}_8$) are used to compute the minimum distance is important in building up the distance map. The algorithm has complexity $O(dN)$, where N is the total number of voxels in the volume.

```
**Input**: $\tilde{\phi}(x, y, z)$, $d$
**Output**: $\phi(x, y, z)$
**for** $i \leftarrow 1$ **to** $d$ **do**
    **for** $x, y, z \in \Omega$ **do**
        **if** $|\tilde{\phi}(x, y, z)| \leq i - 1 + \epsilon$ **then continue**
        **if** $\tilde{\phi}(x, y, z) > 0.0$ **then**                /* Build +ve distances */
            $\alpha_6 \leftarrow \min\{\mathcal{N}_6(\tilde{\phi}(x, y, z))\}$          /* Six face neighbors */
            **if** $\alpha_6 < i$ **then** $\phi(x, y, z) \leftarrow i$; **continue**
            $\alpha_{12} \leftarrow \min\{\mathcal{N}_{12}(\tilde{\phi}(x, y, z))\}$         /* 12 edge neighbors */
            **if** $\alpha_{12} < i$ **then** $\phi(x, y, z) \leftarrow (i + \sqrt{2} - 1)$; **continue**
            $\alpha_8 \leftarrow \min\{\mathcal{N}_8(\tilde{\phi}(x, y, z))\}$     /* Eight vertex neighbors */
            **if** $\alpha_8 < i$ **then** $\phi(x, y, z) \leftarrow (i + \sqrt{3} - 1)$; **continue**
        **else**                                        /* Build -ve distances */
            $\alpha_6 \leftarrow \max\{\mathcal{N}_6(\tilde{\phi}(x, y, z))\}$
            **if** $\alpha_6 > -i$ **then** $\phi(x, y, z) \leftarrow -i$; **continue**
            $\alpha_{12} \leftarrow \max\{\mathcal{N}_{12}(\tilde{\phi}(x, y, z))\}$
            **if** $\alpha_{12} > -i$ **then** $\phi(x, y, z) \leftarrow -(i + \sqrt{2} - 1)$; **continue**
            $\alpha_8 \leftarrow \max\{\mathcal{N}_8(\tilde{\phi}(x, y, z))\}$
            **if** $\alpha_8 > -i$ **then** $\phi(x, y, z) \leftarrow -(i + \sqrt{3} - 1)$; **continue**
        **end**
        $\phi(x, y, z) \leftarrow \tilde{\phi}(x, y, z)$    /* Retain old values outside narrow
        band */
    **end**
**end**
```

Algorithm 6.1: Computation of signed distance field.

Every voxel is updated based on the values of the neighbors. The resulting layer has distance values that are locally Euclidean [74]. The kernel to compute a signed distance layer operates on the computational-volume that has a shared 1-voxel border.

Average values

In a two-domain segmentation, the zero level set of ϕ divides Ω into two regions. Average values c_1 and c_2 can be easily computed over these regions.

Multi-domain segmentation creates more than two regions corresponding to every class of segmentation. We use binary indexing (as explained in section 6.3) to keep track of inside and outside in every implicit function. Thus, for any $q \in [0, n-1]$ indexing a region, we can compute the average value of image intensities.

The computation of average values is a serial operation and requires parsing all the values in the dataset. Reduction algorithms do exist for a parallel computation of sum like operations [18]. A CUDA implementation of the same exists as the CUDPP library by Harris and Sengupta [60]. With CUDPP, however, it is difficult to sum up datasets that cannot fit into the device memory, thus requiring some sort of data slicing. In our experience, the overhead of data slicing, setting up the prefix sum and computing average values turns out to be more expensive than a CPU computation of the averages. Therefore, the average values are computed on the host.

Cubic coefficients

Higher order level set requires a spline representation of the implicit function $\mathbf{\Phi}$. The cubic coefficients are computed for a set of data values sampled along any direction. Tri-cubic spline coefficients can be derived from these coefficients by computing the cubic coefficients along each direction sequentially. The coefficients are derived along an axis in planar sections orthogonal to one of the co-ordinate axes (see Figure 6.3). The computation takes place in three steps, sequentially along each coordinate direction.

The memory manager determines the largest possible sub-volume, the *GPU slice*, that can fit into the available device memory. The full volume is then processed in sizes of the GPU slice. For every plane in the GPU slice, cubic coefficients are computed along the segments parallel to one of the coordinate axes. The resulting coefficients are written to device array of same size as

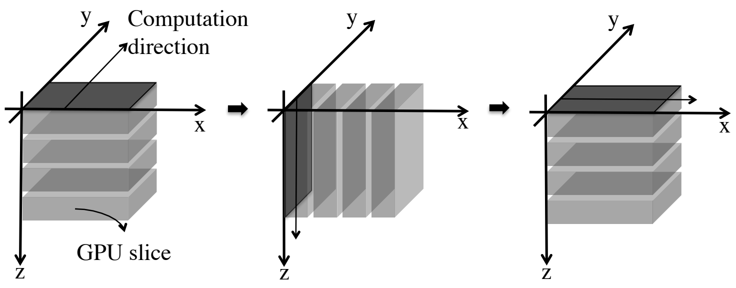


Figure 6.3: *Cubic coefficient computation.*

the GPU slice and copied back to the host array holding the coefficients. The CUDA kernel for computing coefficients works on a linear section per thread. The intermediate sequences are not stored, but recomputed during the recursive process to evaluate c_i in order to reduce the memory footprint of the kernel.

PDE time stepping

The two-phase scalar equation is solved by discretizing the terms of (6.2) using finite differencing. Equation (6.3) is solved by discretizing the lower order term and by computing the spline derivatives for the higher order curvature term. Each PDE has a single higher order term, and 2^{m-1} terms involving average values. Binary indexing is used again to enumerate the lower order terms in every PDE.

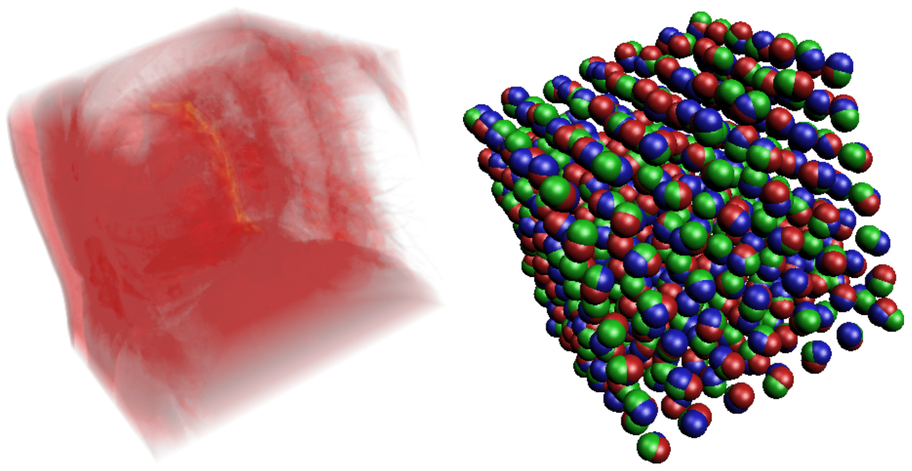
Since all the PDEs must be updated simultaneously (i.e., every voxel in all implicit functions must be updated in parallel), cubic coefficients for all the implicit functions in the device memory are required at all times. This requires that m (that could be arbitrary) arrays for cubic coefficients are stored as 3D texture blocks in the device memory. However, CUDA does not allow dynamically creating texture references. Furthermore, a 4D texture (array of 3D textures) is also not possible in CUDA. Therefore, we simulate a 4D texture by a large 3D texture containing computational-volume sized arrays to hold cubic coefficients for all implicit functions.

There is a possibility of wasting some device memory here for a very large number of implicit functions, since for some m (>16) it might not be possible to have a rectangular 3D array. This is because the largest 3D texture in CUDA can be of size $2^{11} \times 2^{11} \times 2^{11}$, and we hit this limit along one dimension for a computational-volume of size $128 \times 128 \times 128$ and $m > 16$. In such a case, the unusable device memory locked in the texture can be minimized by computing optimal values of three positive factors $m_x, m_y, m_z \in \mathbb{Z}^+$ for $\hat{m} \geq m \in \mathbb{Z}^+$ such that $(\hat{m} - m)$ is minimized and $\hat{m} = m_x \times m_y \times m_z$. The three factors are the number of computational-volumes along the coordinate axes. In doing so, m_x is made as large as possible, followed by a similar heuristic for m_y.

With sub-volumes of $\mathbf{\Phi}$, I and the coefficients cached on the GPU, the PDE is updated for every voxel and for all the implicit functions.

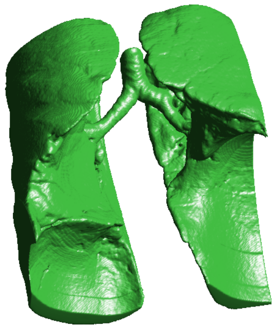
6.5 Results

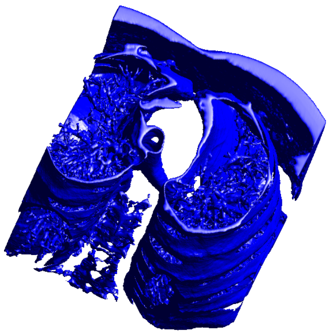
We present results of multi-domain segmentation on different volumetric images, followed by GPU performance statistics. The tests are produced on an Nvidia Tesla C870 device and a Dual-Core AMD Opteron$^{\text{TM}}$ processor 1210. The GPU has 16 multiprocessors (128 processor cores) running at a clock speed of 1.35 GHz and an onboard memory of 1.6 GB. The CPU runs at a clock speed of 1.8 GHz, and has a physical memory of 2 GB.

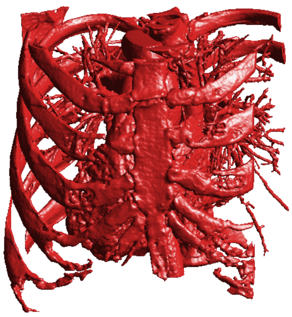


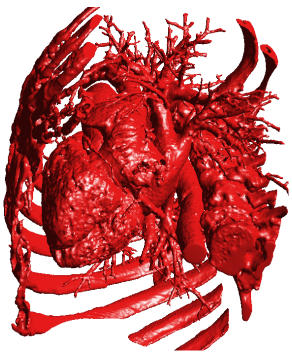
(a) Volume rendering showing various features inside the CT scan (b) Multiphase initialization with spheres for three implicit functions

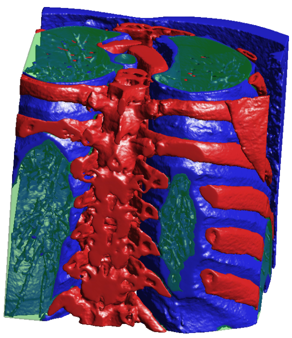
Figure 6.4: *The human thoracic cage.*


(a) Lungs, trachea, and bronchi


(b) Bronchial tree and muscles in front of the rib cage


(c) Ribs and sternum


(d) Heart showing the atrium, the ventricles and the Vena cava


(e) Composite view of the segmented surfaces showing the back bone. A clear separation between the muscles and the ribs can be seen

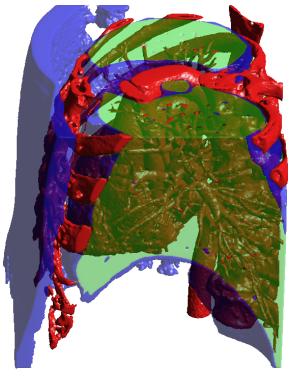
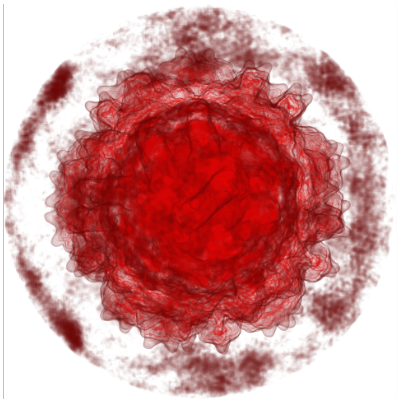

(f) Composite view of the segmented surfaces showing the heart and the blood vessels around it

Figure 6.5: *Multi-domain segmentation of the human thoracic cage. Convergence was declared in 1000 iterations running at 436.8 seconds/iteration and yielding an overall speedup of 5.82 times.*

Figure 6.4(a) shows a volume rendering of a Computed Tomography (CT) scan of the human thoracic cage. The Volume has a size of $512 \times 512 \times 368$ voxels. Segmentation parameters for this volume are: $\lambda_1 = \lambda_2 = 1$, $\mu = .000005 \times 255 \times 255$, $\epsilon = 1$, and $m = 3$. The interfaces for three implicit functions are initialized to a number of spheres (with randomized centers) arranged in a grid (see Figure 6.4(b)). The serial (CPU based implementation) runtime for the segmentation is 2542.0 seconds per iteration, while the parallel (GPU based implementation) runtime reduces to 436.8 seconds per iteration. A total of 1000 iterations with a time step of 0.005 were carried out. Speedups for individual kernels are shown in Figure 6.14. An overall speedup of 5.82 times is achieved for this dataset (indicated with red dotted marker line in Figure 6.14)

The multiphase segmentation results in three prominent regions. The first region consists of the lungs as shown in Figure 6.5(a). The second region consists of the bronchial tree and the muscles in front of the lungs (see Figure 6.5(b)). The last region segments out the bones and the heart. The ribs and the sternum are shown in Figure 6.5(c). The heart segmentation in 6.5(d) clearly shows the atrium, the ventricles, the Vena cova and the blood vessels around he heart. The two distinct features, bones and the heart, are segmented in a single region due to similarity in the intensity of the corresponding voxels in the CT volume. A composite of all the segmented volumes in Figure 6.5(e) shows that the regions are mutually exclusive with no overlap. The backbone can also be clearly seen in the figure. Another composite view of the three regions in Figure 6.5(f) shows the heart, blood vessels, muscles, and the ribs.


(a) Volume rendering showing the capsid and the dsRNA

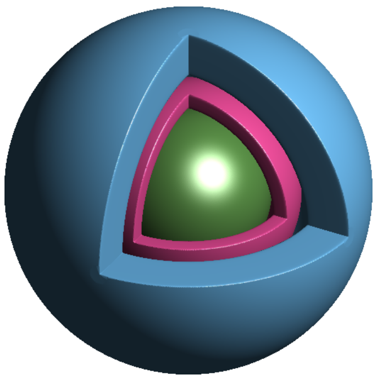
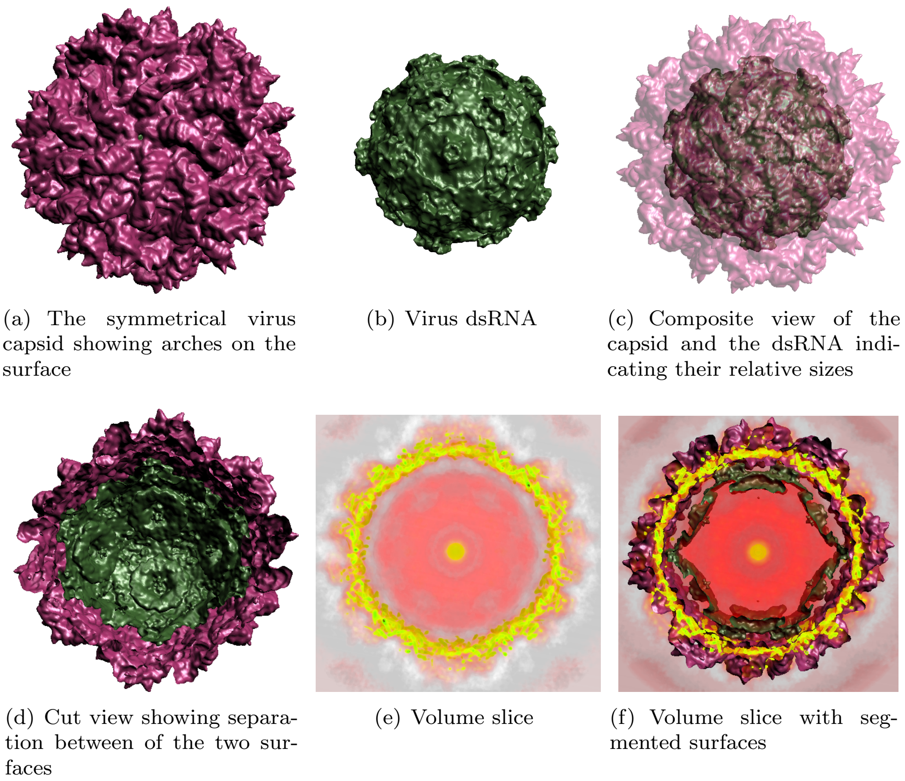

(b) Multiphase initialization for three implicit functions (cut through section)

Figure 6.6: *Penicillium Stoloniferum virus (PSV).*

The next example we consider is segmentation of the Penicillium Stoloniferum virus (PSV), a virus that infects the fungus that makes Penicillin. Figure 6.6(a) shows a volume rendering of the PSV. The dataset shown here is a cryo-electron microscopy (cryoEM) image at 7.35Å resolution. The volume size is $381 \times 381 \times 381$ voxels. Interface initialization for the virus for three implicit functions consisted of a sphere for the double stranded RNA (dsRNA), a spherical shell for the capsid and another spherical shell for the outer matter respectively (see Figure 6.6(b)). The level set parameters are kept the same as before. The serial runtime for the segmentation is 1673.9 seconds per iteration, while the parallel runtime reduces to 267.4 seconds per iteration. A total of 300 iterations with a time step of 0.005 were carried out. Speedups for individual kernels are shown in Figure 6.14. An overall speedup of 6.26 times is achieved for this dataset (indicated with red dotted marker line in Figure 6.14). The reason for faster convergence in this case is attributed to the modified initialization of the interfaces suited to the features present in the PSV volume data.


(a) The symmetrical virus capsid showing arches on the surface

(b) Virus dsRNA

(c) Composite view of the capsid and the dsRNA indicating their relative sizes

(d) Cut view showing separation between of the two surfaces

(e) Volume slice

(f) Volume slice with segmented surfaces

Figure 6.7: *Penicillium Stoloniferum virus (PSV) segmentation. Convergence was declared in 300 iterations running at 267.4 seconds/iteration and yielding an overall speedup of 6.26 times.*

The segmentation results in the symmetrical capsid (see Figure 6.7(a)) showing a local 2-fold symmetry forming prominent surface arches. The dsRNA of the virus also has a prominent symmetrical structure with projections on the surface (see Figure 6.7(b)). The implicit function corresponding to the outer region segments the remaining volume of the virus (not shown here), which is complementary to the union of the two other segmented regions. Figure 6.7(c) shows a composite view of the capsid and the dsRNA indicating the relative sizes of the two segments. Shown in Figure 6.7(d) is a cut through section of the two surfaces showing a clear separation between the two. Figure 6.7(e) shows a slice of the volume highlighting the density variations inside the scanned volume. Figure 6.7(f) overlays the extracted surfaces on top of the volume slice to show details of the dsRNA and the capsid.

6.5.1 Reconstruction accuracy

We analyze the reconstruction accuracy with a synthetic example of a phantom volume of size $128 \times 128 \times 128$ consisting of a CSG (Constructive Solid geometry) object formed by the union of a cuboid and a sphere as shown in Figure 6.8. Additive Gaussian noise of specific standard deviations are added to the clean volume (see Figures 6.9(a), 6.10(b), 6.11(a), and 6.12(a)). The reconstructed objects are shown in Figures 6.9-6.12 along with the distribution of the Hausdorff distance on the model surface. The vertical bar alongside the distribution shows the range of the distance values (absolute) and their histogram. We measure the reconstruction error by means of the *symmetrical Hausdorff distance* which is a good measure of the distance between two meshes (see Aspert et al. [6]).

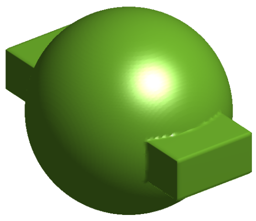
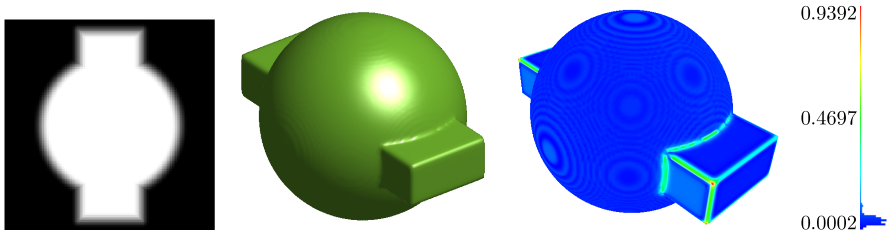


Figure 6.8: *The CSG Phantom object consisting of a sphere and a cuboid.*

Symmetrical Hausdorff distance, d_H, between two surfaces M_0 and M_1 is given by

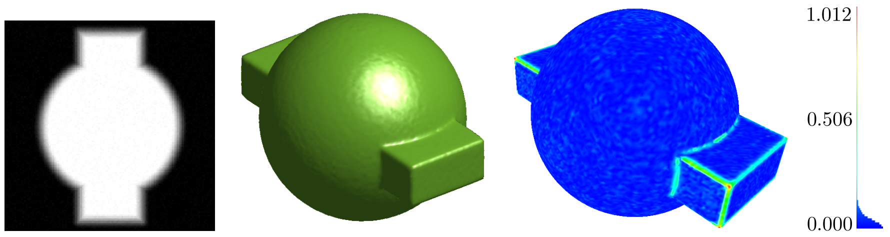
$$d_H(M_0, M_1) = \max\left\{ \sup_{x_0 \in M_0} \inf_{x_1 \in M_1} d(x_0, x_1), \sup_{x_1 \in M_1} \inf_{x_0 \in M_0} d(x_0, x_1) \right\}, \quad (6.8)$$

where $d(\cdot, \cdot)$ is an appropriate metric for measuring distance between two points in a metric space. $d_H(M_0, M_1)$ measures the maximum possible distance that



(a) A cross section of the volume (b) Reconstruction surface (c) Distribution of the Hausdorff distance on the phantom surface

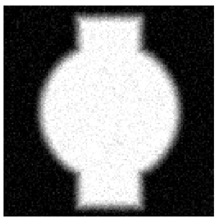
Figure 6.9: *Reconstruction from phantom image.*

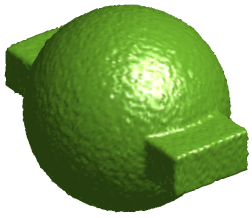


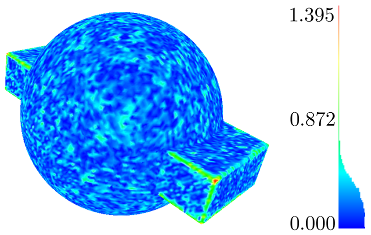
(a) A cross section of the volume (b) Reconstruction surface (c) Distribution of the Hausdorff distance on the phantom surface

Figure 6.10: *Reconstruction from phantom image with added Gaussian noise of* $\sigma = 0.01$.

will be required to travel from surface M_0 to M_1. We compute this metric and use it to quantify the error in reconstruction of a surface from noisy volume. Table. 6.1 shows maximum, mean and RMS d_H for increasing noise (zero mean) in the volumes with standard deviations of 0.00, 0.01, 0.05, and 0.10 voxel units respectively. In the table, BBox% refers to the relative Hausdorff distance measured as the percentage of the model bounding box diagonal. From the results, the accuracy of reconstruction (considering mean and RMS d_H) decreases with increasing noise, nevertheless it always remains below a relative d_H of 0.2% of the bounding box diagonal of the object. The results presented here are produced without any filtering on the synthetic volumes. For very high noise content in the images, a preprocessing stage such as median filtering or anisotropic filtering is generally recommended that smears out the noise.

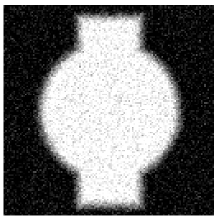

(a) A cross section of the volume

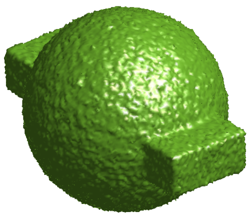

(b) Reconstruction surface

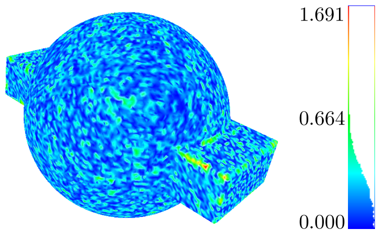


(c) Distribution of the Hausdorff distance on the phantom surface

Figure 6.11: *Reconstruction from phantom image with added Gaussian noise of* $\sigma = 0.05$.


(a) A cross section of the volume


(b) Reconstruction surface



(c) Distribution of the Hausdorff distance on the phantom surface

Figure 6.12: *Reconstruction from phantom image with added Gaussian noise of* $\sigma = 0.10$.

6.5.2 Speedup

The speedup of a parallel program is defined as the ratio of the time taken to solve a problem on a single processing element to the time required to solve the same problem on a parallel computer with p processors [55]. Denoting by T_s the serial runtime and by T_p the parallel runtime, the speedup S is given by

$$S = \frac{T_s}{T_p}. \tag{6.9}$$

For a given problem if there exist parts of the algorithm that cannot be paral-

Table 6.1: *Reconstruction error in noisy volumes.*

d_H		Noise ($\mu = 0$) standard deviation σ			
		0.00	0.01	0.05	0.10
Max	Absolute	0.9392	1.0117	1.3946	1.6909
	(BBox %)	(0.565 %)	(0.608 %)	(0.836 %)	(1.001 %)
Mean	Absolute	0.0509	0.0457	0.1814	0.2605
	(BBox %)	(0.031 %)	(0.027 %)	(0.109 %)	(0.154 %)
RMS	Absolute	0.0786	0.0804	0.2100	0.3041
	(BBox %)	(0.047 %)	(0.048 %)	(0.126 %)	(0.180 %)

lelized, then these must also be considered in T_p. Thus, for a given problem if α is the fraction of algorithm that cannot be parallelized, then the speedup can be written as

$$S = \frac{T_s}{\alpha T_s + (1-\alpha)T_s/p} = \frac{p}{1+\alpha(p-1)}. \tag{6.10}$$

This expression of S results from the so called *Gustafson's law* [57]. As α becomes smaller, S becomes close to p. In case of GPU computing, an expression for S becomes more complicated and many other factors come into play. Global memory access is rather slow in the whole GPU computation pipeline and introduces delays in the application. Thread synchronization also adds to the delay by idling the threads. At an application level, the memory manager is subjected to the overheads of blocking the full volume into sub volumes and of data transfers to and from the device. This overhead can be reduced by choosing a larger blocking size provided the device has more onboard memory. In order to evaluate the performance of our solver, we calculate speedups of the individual kernels and the whole application.

We use datasets of different sizes to quantify speedups with our CUDA based streaming solver. A total of five datasets were used to benchmark both the tri-linear and the tri-cubic level set segmentation. The runtimes are averaged over ten iterations of each run. the dataset sizes are indicated in Figures 6.13 and 6.14. In increasing order of size of the volumes in voxels, the datasets that we used for the benchmark tests are the brain CT image of size $512 \times 512 \times 28$, cryo-EM image of the PSV at 7.3Å resolution and size $381 \times 381 \times 381$, cryo-EM image of the Reovirus at 6.6Å resolution and size $403 \times 403 \times 403$, CT scan of the human thoracic region of size $512 \times 512 \times 368$, and electron tomography image of the Simian Immunodeficiency Virus (SIV) of size $830 \times 950 \times 200$. Due to the streaming nature of the solver, the computational resolution of the

CUDA kernels is limited to the size of the computational volume. Therefore, marginally more computations are performed by the GPU solver compared to a CPU implementation. For example, for a computational volume of size 128^3 and the PSV volume of size $381 \times 381 \times 381$, the number of blocks of computational volumes processed in every solver iteration is $64(= 4 \times 4 \times 4)$. It must be noticed that the computational volume also includes a 1-voxel border used for padding. For a slightly larger volume of size $403 \times 403 \times 403$ of the Reovirus data, the number of computational volumes processed by the solver are also 64, thus resulting in a similar runtime as with the PSV data. However, the CPU runtimes are different for these two.

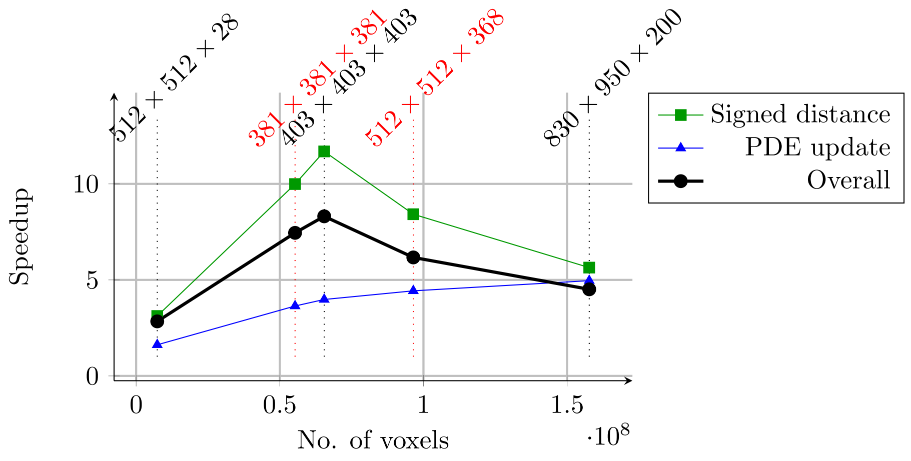


Figure 6.13: *GPU speedup for tri-linear segmentation. Specific volume sizes used for the benchmark are indicated at the top (marked in red are the human thoracic case and the PSV datasets).*

Interface initialization CUDA kernel has low arithmetic intensity, therefore very high speedups from 35 times to 92 times are achieved for different sized volumes. Signed distance computation on the GPU yields speedups in the range 3-12 times. On the other hand, PDE updates are expensive in terms of arithmetic operations, thus giving an average speedup of about 3.7 times with tri-linear update and that of 10 times with tri-cubic update. It should be noted that the tri-cubic PDE update is faster than the tri-linear one since the later uses finite differencing to compute higher order derivatives, while the earlier uses texture lookups and fewer computations. Cubic coefficients are expensive to compute, thus yielding an average speedup of about 1.5 times. We must note here that none of the kernels hit the memory limit on the device, which is attributed to the streaming nature of the solver and an appropriately chosen subvolume size.

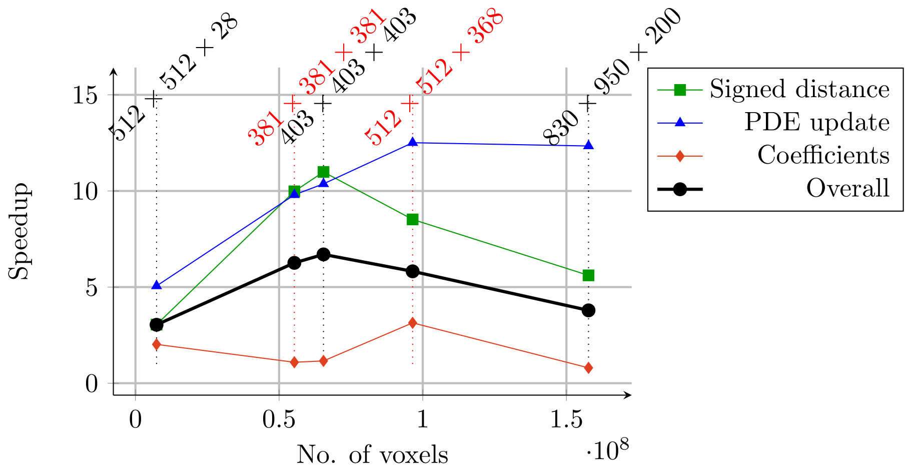


Figure 6.14: *GPU speedup for tri-cubic segmentation. Specific volume sizes used for the benchmark are indicated at the top (marked in red are the human thoracic case and the PSV datasets).*

Performance speedup graphs of GPU computations compared to CPU ones are shown in Figures 6.13 and 6.14. Speedup curves for interface initialization, which have very high values, are not shown for sake of clarity of other curves. The shown speedup curves show a single peak for varying volume size. GPU performance is a complicated phenomenon and various factors contribute to the final speedup. With an increase in the problem size, the increase in speedup increases until a point, thus reaching a limit of the speedup. Various factors such as excessive memory paging, and increased block transfer overhead come into play and decrease the rate of change of speedup. For a larger problem, the overhead of memory transfer increases with the cube of the increase in size. Since the device computation reads the volume in small chunks of memory, this access pattern hits the performance even more by increasing the number of page faults. Nevertheless, the overall speedup, computed using (6.10), is always greater than three. Average overall performance for the tri-linear case is 5.8 times and for the tri-cubic case is 5.1 times.

6.6 Conclusions

In this work we presented a framework for streaming computations on the GPU. The framework is employed for efficient computation of the multi-domain, and higher order level set method applied to the Mumford-Shah energy functional. The presented framework is generic and can be easily used with other energy functional as well. We show results of the segmentation on two different imaging modalities along with performance speedups obtained with the solver. The overall performance gain obtained is about 6 times for the two-domain segmentation and about 5 times for the multi-domain segmentation. The solver has been implemented in the freely available UT-CVC image processing and visualization software called VolRover [135] that can be downloaded at `http://cvcweb.ices.utexas.edu/cvc/projects/project.php?proID=9`.

Acknowledgements

This work was done while Ojaswa Sharma was visiting Chandrajit Bajaj at UT-CVC. His visit was supported by the Technical University of Denmark. The research of Qin Zhang and Chandrajit Bajaj was supported in part by NSF grant CNS-0540033 and NIH contracts R01-EB00487, R01-GM074258.

Chapter 7

Homotopy based surface reconstruction with application to acoustic signals

O. Sharma[†], and F. Anton[†]

[†]Department of Informatics and Mathematical Modelling,
Technical University of Denmark, Lyngby, 2800, Denmark

Abstract

This work introduces a new algorithm for surface reconstruction in $\mathbb{R}^3$ from spatially arranged one-dimensional cross sections embedded in $\mathbb{R}^3$. This is generally the case with acoustic signals that pierce an object non-destructively. Continuous deformations (homotopies) that smoothly reconstruct information between any pair of successive cross sections are derived. The zero level set of the resulting homotopy field generates the desired surface. Four types of homotopies are suggested that are well suited to generate a smooth surface. We also provide derivation of necessary higher order homotopies that can generate a C^2 surface. An algorithm to generate surface from acoustic sonar signals is presented with results. Reconstruction accuracies of the homotopies are compared by means of simulations performed on basic geometric primitives.

7.1 Introduction

Surface reconstruction is a frequently encountered problem in computer graphics and computer vision. The reconstruction problem that we address in this paper is the one of generating a topologically and geometrically convincing surface from a set of acoustic signals acquired using multi-beam echo-sounders.

The problem of object reconstruction from cross sections is quite old and has been addressed in different forms. A lot of work has been done in 3D object reconstruction from planar cross sections (see, for example, [70, 47, 19, 12]). The cross sections considered in these are generally contours of the objects that could be parallel or non-parallel as discussed by Boissonnat and Memari [20], and later by Liu et al. [83]. Hoppe et al. [61] discuss the problem of object reconstruction from a point cloud which is also of widespread interest. This problem has been analyzed from different perspectives. For example, Carr et al. [22] use radial basis functions for reconstruction. Amenta and Bern [4] use Voronoi filtering to generate surface from point clouds.

In the context of reconstruction from acoustic signals, most of the work focusses on reconstruction from cross section images (see, for example, the work by Zhang et al. [146]). In fact, acoustic images are obtained by interpolating intensities from planar acoustic beams arranged in a fan. A better algorithm can be designed to reconstruct the underlying object from original signals without relying on a simple interpolation based estimate.

Homotopy continuation is a powerful mathematical tool for robustly solving a complex system of equations (see Allgower and Georg [3]). Continuation based method suggested for surface reconstruction from planar contours by Shinagawa and Kunii [124] uses a straight line homotopy to generate smooth surface. Their method generates a minimal surface by finding optimal path in the toroidal graph representation. Berzin and Hagiwara [17] analyze minimal area criterion in surface reconstruction using homotopy and show that such criteria lead to defective surfaces. An isotopy based reconstruction scheme is proposed by Fujimura and Kuo [48], in which bifurcations are handled separately. In this paper we present a different reconstruction algorithm that utilizes continuous deformations of functions for tracing the reconstruction boundary. We develop homotopies other than linear homotopies that can generate smooth surface. The homotopies developed here are built to take advantage of the spatial arrangement of the signals.

This paper is organized as follows. In section 7.2 we review the basic homotopy theory. Section 7.3 presents a brief overview of the acoustic signals from which a reconstruction is desired. Section 7.4 outlines our approach to reconstruction using homotopy deformation. We develop various homotopies suited to reconstruction. Section 7.5 is devoted to the reconstruction algorithm utilizing the concepts developed so far. In sections 7.6 and 7.7 we present the results and time complexity of the presented algorithm, respectively. We conclude the discussion in section 7.8.

7.2 Homotopy continuation

The central idea of the reconstruction algorithm presented here is *homotopy* (see Armstrong [5]) or *continuous deformation.* Two continuous functions $f_0(\mathbf{x})$ and $f_1(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^N$, are called homotopic if one can be continuously deformed into the other. Such a deformation is called a homotopy $\mathcal{H}(\mathbf{x}, \lambda)$ (in parameter $\lambda \in \mathbb{R}$) between the two functions.

In other words, a family of continuous mappings

$$h_\lambda : X \mapsto Y, \lambda \in [0,1] \tag{7.1}$$

is called a homotopy if the function

$$\mathcal{H} : X \times [0,1] \mapsto Y \tag{7.2}$$

defined by

$$\mathcal{H}(\mathbf{x}, \lambda) = h_\lambda(\mathbf{x}), \mathbf{x} \in X, \lambda \in [0,1] \tag{7.3}$$

is continuous. The maps h_0 and h_1 are called the *initial map* and the *terminal map* of the homotopy h_λ. A typical choice is a *linear homotopy* such as

$$\mathcal{H}(\mathbf{x}, \lambda) = (1-\lambda) f_0(\mathbf{x}) + \lambda f_1(\mathbf{x}). \tag{7.4}$$

The use of deformations to solve non-linear system of equations gives robust results. A homotopy tries to solve a difficult problem with unknown solutions by starting with a simple problem with known solutions. Stable predictor-corrector and piecewise-linear methods for solving such problems exist (see Allgower and Georg [3]). The system $\mathcal{H}(\mathbf{x}, \lambda) = 0$ implicitly defines a curve or one-manifold of solution points.

Given smooth $\mathcal{H}$ and existence of $u_0 \in \mathbb{R}^{N+1}$ such that $\mathcal{H}(u_0) = 0$ and $\text{rank}(\mathcal{H}'(u_0)) = N$, there exists a smooth curve $\alpha \in J \mapsto c(\alpha) \in \mathbb{R}^{N+1}$ for some open interval J containing zero such that for all $\alpha \in J$ (Allgower and Georg [3])

1. $c(0) = u_0$,
2. $\mathcal{H}(c(\alpha)) = 0$,
3. $\text{rank}(\mathcal{H}'(c(\alpha))) = N$,
4. $c'(\alpha) \neq 0$.

The map $\mathcal{H}$ deforms f_0 to f_1 in a smooth fashion via the path c.

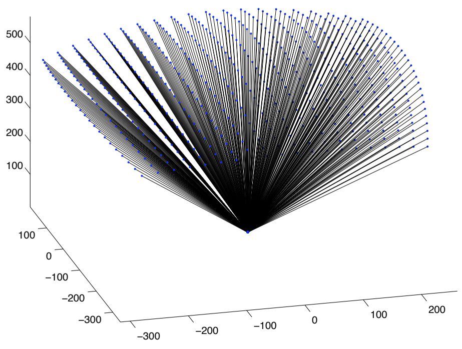
We use these concepts in section 7.4 where we define non-linear, spline and shape preserving homotopies. Next section introduces the nature of acoustic signals and their spatial arrangement.

7.3 Acoustic signals

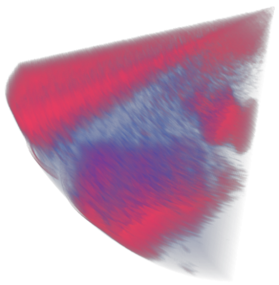
Multibeam sonar data acquisition results in huge amount of data in small time. The data is in the form of multiple beams of signals that are arranged in a particular geometry for an instrument. The MS70 Multibeam echo-sounder from Simrad is a 3D sonar where a total of 500 beams are arranged in a fashion such that an angular cone of 45° by 60° is spanned by a matrix of 20×25 beams. The echo-sounder operates on a frequency range of 75 to 112 kHz (see Ona et al. [99] for details).

Multibeam echo-sounders sample the space non uniformly since the linear spacing between individual beams increases with distance along the beams. As a result, the objects far away from the instrument have a very coarse resolution in the sampled volume. Figure 7.1(a) shows a typical arrangement of the MS70 echo-sounder. A volume rendering of a single ping capturing a moving school of Sprat fish is shown in Figure 7.1(b).

In order to reconstruct a surface representation of the objects imaged by the sonar, we formulate homotopies for beams. These are discussed in the next section.



(a) MS70 beam configuration



(b) Volume rendering of sample data (high intensities in red)

Figure 7.1: *The MS70 echo-sounder.*

7.4 Homotopic reconstruction

Consider a signal S and its piecewise constant representation G. In other words, G is a segmentation of S with classes (or levels) $C_0, C_1, \cdots,$ and C_{n-1}. Let us denote by χ_k the characteristic function of S for class C_k, such that

$$\chi_k = \begin{cases} 1 & \text{if } G = C_k, \\ 0 & \text{otherwise.} \end{cases} \tag{7.5}$$

The main idea behind reconstruction for any level C_k using continuous deformations is to trace the path $\mathbf{c} = \ker(\mathcal{H}) = \{(\mathbf{x}, \lambda) : \mathcal{H} = 0\}$ between functions defined on any two consecutive signals (as shown in Figure 7.2 in $\mathbb{R}^2$).

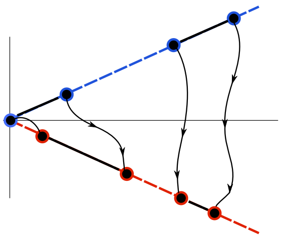


Figure 7.2: *Homotopy path between two beams.*

In order to be able to define homotopies between pairs of beams, we need to associate functions with each beam. Such a *beam function* should completely describe the boundary, interior and exterior of regions belonging to any class C_k along the respective beam. These regions are intervals in one-dimensional cross sections. The rest of the discussion applies to any class, therefore we drop the subscript k for sake of simplicity of notation. Given roots $r_i, i \in [0, p-1]$ of the characteristic function χ of a beam, we define its beam function as a piecewise polynomial. This can be done by selecting slopes at the roots of the desired piecewise polynomial and enforcing continuity at borders of adjacent polynomials. The simplest piecewise polynomial exhibiting C^1 continuity is a piecewise quadratic

$$f(r) = \sum_{i=0}^{p-2} \frac{\alpha_i(-r^2 + r(r_i + r_{i+1}) - r_i r_{i+1}))}{(r_{i+1} - r_i)}, \tag{7.6}$$

where r is the distance along the beam, and α_i is the positive gradient $|\frac{\mathrm{d}f}{\mathrm{d}r}|_{r=r_i}$ defined as

$$\alpha_i = (-1)^{i+1}\alpha_0, \tag{7.7}$$

with α_0 being a chosen positive slope at r_0. A quadratic polynomial also keeps the system simple to solve. Figure 7.3 illustrates such a beam function.

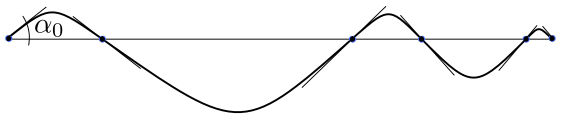


Figure 7.3: *Piecewise quadratic function representing a beam.*

With this background, we define various homotopies in the following subsections.

7.4.1 Linear homotopy

Consider beam functions $f_j(r)$ and $f_{j+1}(r)$ for two consecutive beams with angles θ_j and θ_{j+1} respectively in the polar plane (r, θ). We define a homotopy $\mathcal{H} : \mathbb{R}^2 \mapsto \mathbb{R}$ of smooth transition from $f_j(r)$ to $f_{j+1}(r)$ using a real parameter $\lambda \in [0, 1]$ as

$$\mathcal{H}_j(r, \lambda) = (1 - \lambda)f_j(r) + \lambda f_{j+1}(r), \tag{7.8}$$

where the parameter λ is related to the angle θ as

$$\lambda = \frac{\theta - \theta_j}{\theta_{j+1} - \theta_j}. \tag{7.9}$$

Such an $\mathcal{H}$ is a linear homotopy that transforms $f_j(r)$ to $f_{j+1}(r)$ using a linear combination of these functions in parameter λ. We can now define a set of homotopies that reconstructs the underlying object from a set of beams as the set $\mathbf{H} = \{\mathcal{H}_j\}$ where each homotopy $\mathcal{H}_j$ is defined for a pair of beam signals S_j and S_{j+1}.

A reconstruction from $\mathbf{H}$ is then given by the curve $\mathbf{c} = \ker(\mathbf{H}) = \bigcup_{j \in J} \ker(\mathcal{H}_j)$. Figure 7.4 shows part of the reconstruction of a set of radial beams.

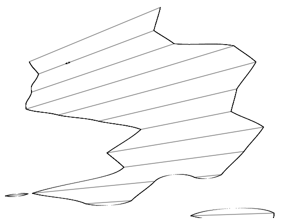


Figure 7.4: *Reconstruction using linear homotopy. Reconstructed object (with more than one connected components) boundary connects the end points of radial cross sections (in gray).*

Proposition 7.1 $\mathbf{H}$ *defined in (7.8) results in a piecewise non-linear curve* $\mathbf{c}$ *that is only* C^0 *in* θ.

Proof. We prove this by showing that

1. $\mathbf{c}$ is C^0 at (r_i, θ_{j+1})[1], and
2. $\mathbf{c}$ is not C^1 at (r_i, θ_{j+1}).

For 1), it is sufficient to show that $\mathcal{H}_j(r_i, \theta_{j+1}) = \mathcal{H}_{j+1}(r_i, \theta_{j+1})$. From (7.8)

$$\begin{aligned} \mathcal{H}_j(r_i, \theta_{j+1}) &= f_{j+1}(r_i), \text{and} \\ \mathcal{H}_{j+1}(r_i, \theta_{j+1}) &= f_{j+1}(r_i). \end{aligned}$$

Therefore, $\mathbf{c}$ is C^0.

[1]This point belongs to $\mathbf{c}$ since r_i is a zero of $f_{j+1}(r)$ by construction.

For 2), consider the tangent at any point (r, θ) for $\mathcal{H}_j$

$$\begin{aligned} T_j &= \left(\frac{\partial \mathcal{H}_j}{\partial r}, \frac{\partial \mathcal{H}_j}{\partial \theta}\right) \qquad (7.10) \\ &= \left((1-\lambda)f_j'(r) + \lambda f_{j+1}'(r), \frac{-f_j(r) + f_{j+1}(r)}{\triangle\theta_j}\right), \end{aligned}$$

where $\triangle\theta_j = (\theta_{j+1} - \theta_j)$. At $r = r_i$, we note that

$$\begin{aligned} \lim_{\theta \to \theta_{j+1}^-} T_{j+1} &= \left(f_{j+1}'(r_i), \frac{-f_j(r_i) + f_{j+1}(r_i)}{\triangle\theta_j}\right), \text{and} \qquad (7.11) \\ \lim_{\theta \to \theta_{j+1}^+} T_{j+1} &= \left(f_{j+1}'(r_i), \frac{-f_{j+1}(r_i) + f_{j+2}(r_i)}{\triangle\theta_{j+1}}\right). \end{aligned}$$

Therefore, $\lim_{\theta \to \theta_{j+1}^-} T_{j+1} \neq \lim_{\theta \to \theta_{j+1}^+} T_{j+1}$, and $\mathbf{c}$ is not C^1. This shows that $\mathbf{c}$ is only C^0 in θ. □

Therefore, we seek a homotopy that preserves tangent slopes at the joins. This leads us to the introduction of *non-linear homotopy.*

7.4.2 Non-linear homotopy

The curve $\mathbf{c} = ker(\mathbf{H})$ resulting from (7.8) is a piecewise smooth curve in $\mathbb{R}^2$. We are interested in at least a C^1 continuous curve generated by a homotopy function. Let's consider the following non-linear homotopy in λ

$$\mathcal{H}_j(r, \lambda, \eta) = (1-\lambda)^\eta f_j(r) + \lambda^\eta f_{j+1}(r). \qquad (7.12)$$

Figure 7.5 shows a reconstruction using the non-linear homotopy defined by (7.12).

Proposition 7.2 *For $\eta > 1$, $ker(\mathbf{H})$ generates at least a C^1 curve in θ for constant angular spacing of beams.*

Proof. We prove this by showing that

1. $\mathbf{c}$ is C^0 at (r_i, θ_{j+1}), and
2. $\mathbf{c}$ is C^1 at (r_i, θ_{j+1}) for uniform angular spacing of beams.

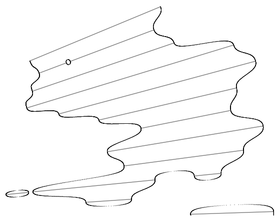


Figure 7.5: *Reconstruction using non-linear homotopy ($\eta = 2$) from the same set of radial cross sections. Staircase effect is prominent in this reconstruction.*

As before we can prove 1) using (7.8) by showing that $\mathcal{H}_j(r_i, \theta_{j+1}) = \mathcal{H}_{j+1}(r_i, \theta_{j+1})$.

For 2), we again consider the tangent at any point (r, θ) for $\mathcal{H}_j$

$$\begin{aligned} T_j &= \left(\frac{\partial \mathcal{H}_j}{\partial r}, \frac{\partial \mathcal{H}_j}{\partial \theta} \right) \\ &= \Bigg(\ (1-\lambda)^\eta f_j'(r) + \lambda^\eta f_{j+1}'(r), \\ &\qquad \frac{-\eta(1-\lambda)^{\eta-1} f_j(r) + \eta \lambda^{\eta-1} f_{j+1}(r)}{\triangle \theta_j} \Bigg) . \end{aligned}$$

At $r = r_i$, we note that

$$\begin{aligned} \lim_{\theta \to \theta_{j+1}^-} T_{j+1} &= \left(f_{j+1}'(r_i), \frac{\eta f_{j+1}(r_i)}{\triangle \theta_j} \right), \text{and} \\ \lim_{\theta \to \theta_{j+1}^+} T_{j+1} &= \left(f_{j+1}'(r_i), \frac{-\eta f_{j+1}(r_i)}{\triangle \theta_{j+1}} \right) . \end{aligned} \tag{7.13}$$

If $\triangle \theta = \triangle \theta_j = \triangle \theta_{j+1}$ is the constant angular spacing between the beams, then

$$\lim_{\theta \to \theta_{j+1}^-} T_{j+1} = \lim_{\theta \to \theta_{j+1}^+} T_{j+1} .$$

This shows that $\mathbf{c}$ is at least C^1 in θ. Further, it can also be shown that the slope at beam end is normal to the radial line at angle θ_{j+1}. This is left as an exercise to the reader. $\square$

A non-linear homotopy is a general case of the linear homotopy. This class of deformations can be extended to higher dimensions in a straightforward manner.

A formulation in $\mathbb{R}^3$ for an arrangement of beams shown in Figure 7.6 can be made as a two parameter homotopy in α and β as

$$\begin{aligned}\mathcal{H}_{j,k}(r,\alpha,\beta,\eta,\zeta) &= f_{j,k}(r)(1-\alpha)^\eta(1-\beta)^\zeta + \\ &\quad f_{j,k+1}(r)(1-\alpha)^\eta\beta^\zeta + \\ &\quad f_{j+1,k}(r)\alpha^\eta(1-\beta)^\zeta + \\ &\quad f_{j+1,k+1}(r)\alpha^\eta\beta^\zeta, \end{aligned} \tag{7.14}$$

where α and β are linearly related to the inclination θ and azimuth ϕ in a spherical coordinate system (r,θ,ϕ) (similar to (7.9) in $\mathbb{R}^2$). The two-parameter homotopy (7.14) can also be written as a tensor product of one-parameter homotopies.

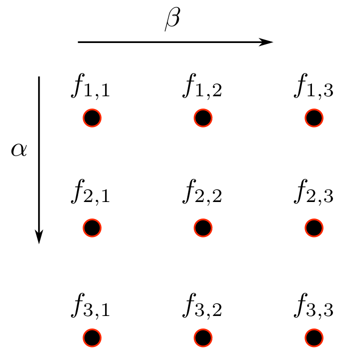


Figure 7.6: *Top view of beam arrangement in $\mathbb{R}^3$. Each dot represents a beam orthogonal to the plane of the paper.*

For the $\mathcal{H}_{j,k}$ defined above, it can be shown that the surface $\mathcal{H} = 0$ is C^1 continuous for $\eta > 1, \zeta > 1$. For $\eta = 1$ and $\zeta = 1$, $\mathcal{H}_{j,k}$ reduces to a linear homotopy.

A non-linear homotopy is continuous at joins and satisfies all the required criteria, however the reconstruction looks unnatural due to the fact that the tangent at the joins are always orthogonal to the respective beams (see Figure 7.5). This constrains the solution to a small class of possible reconstructions. In the next subsection, we relax the tangent constraint that gives rise to the *cubic spline homotopy.*

7.4.3 Cubic spline homotopy

Consider N radial beam functions $\{f_j(r)\}$, $j \in [0, N-1]$ in the polar plane (r, θ). While deriving the necessary conditions, we replace the local parameter λ by its global counterpart θ. These two are related by (7.9). We construct homotopies $\mathcal{H}_j$ between f_j and f_{j+1} such that following conditions are met:

1. For any homotopy, the initial and terminal maps are satisfied

$$\begin{aligned} \mathcal{H}_j(r, \theta_j) &= f_j(r), \text{and} \\ \mathcal{H}_j(r, \theta_{j+1}) &= f_{j+1}(r), \end{aligned} \tag{7.15}$$

   for $j \in [0, N-2]$.

2. The first derivatives $\partial \mathbf{H}(r, \theta)/\partial\theta$ at the boundary of any two successive homotopies match

$$\lim_{\theta \to \theta_{j+1}^-} \frac{\partial \mathcal{H}_j(r, \theta)}{\partial \theta} = \lim_{\theta \to \theta_{j+1}^+} \frac{\partial \mathcal{H}_{j+1}(r, \theta)}{\partial \theta}, \tag{7.16}$$

   for $j \in [0, N-3]$.

3. The second derivatives $\partial^2 \mathbf{H}(r, \theta)/\partial\theta^2$ at the boundary of any two successive homotopies match

$$\lim_{\theta \to \theta_{j+1}^-} \frac{\partial^2 \mathcal{H}_j(r, \theta)}{\partial \theta^2} = \lim_{\theta \to \theta_{j+1}^+} \frac{\partial^2 \mathcal{H}_{j+1}(r, \theta)}{\partial \theta^2}, \tag{7.17}$$

   for $j \in [0, N-3]$.

We start with a general cubic homotopy in θ of the form

$$\mathcal{H}_j(r, \theta) = \sum_{i=0}^{3} g_{j,i}(r)(\theta - \theta_j)^i, \tag{7.18}$$

with unknown coefficient functions $g_{j,i}$. We note the following partial derivatives of $\mathcal{H}_j(r, \theta)$ w.r.t θ

$$\frac{\partial \mathcal{H}_j(r, \theta)}{\partial \theta} = \sum_{i=1}^{3} i g_{j,i}(r)(\theta - \theta_j)^{i-1}, \text{and} \tag{7.19}$$

$$\frac{\partial^2 \mathcal{H}_j(r, \theta)}{\partial \theta^2} = \sum_{i=2}^{3} i(i-1) g_{j,i}(r)(\theta - \theta_j)^{i-2}. \tag{7.20}$$

Note that in the above conditions, we are not concerned with the derivatives of $\mathbf{H}$ w.r.t. r since these have no influence on continuity of $\mathbf{c}$ w.r.t. θ. Continuity of $\mathbf{c}$ w.r.t. r depends on the choice of beam functions $f(r)$. Conditions (7.15), (7.16) and (7.17) result in the following linear system (argument r of g's and f's is removed for space consideration):

$$\begin{aligned} g_{j,0} &= f_j, \text{for} \\ & \quad j \in [0, N-2], \\ g_{j,0} + g_{j,1}\triangle + g_{j,2}\left(\triangle\theta_j\right)^2 + g_{j,3}\left(\triangle\theta_j\right)^3 &= f_{j+1}, \text{for} \\ & \quad j \in [0, N-2], \\ g_{j,1} + 2g_{j,2}\triangle\theta_j + 3g_{j,3}\left(\triangle\theta_j\right)^2 &= g_{j+1,1}, \text{for} \\ & \quad j \in [0, N-3], \\ g_{j,2} + 3g_{i,3}\triangle\theta_j &= g_{j+1,2}, \text{for} \\ & \quad j \in [0, N-3]. \end{aligned} \tag{7.21}$$

Since, system (7.21) is devoid of two conditions, we enforce free boundary condition $\partial^2\mathcal{H}(r,\theta)/\partial\theta^2 = 0$ at θ_0 and θ_{N-1}. This yields the following two linear equations

$$\begin{aligned} g_{0,2} &= 0, \text{and} \\ g_{N-2,2} + 3g_{N-2,3}\triangle\theta_j &= 0. \end{aligned} \tag{7.22}$$

The system formed by (7.21) and (7.22) has the form $A\mathbf{x} = B$. The coefficients $g_{j,i}$ are functions of f_j. The homotopy (7.18) can be rewritten in terms of the local homotopy variable λ. Figure 7.7 shows a reconstruction using the developed cubic spline homotopy. It is possible to extend the single parameter cubic spline homotopy to a two parameter spline homotopy for a reconstruction in $\mathbb{R}^3$ via the tensor product.

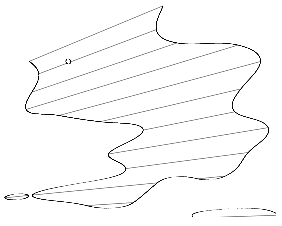


Figure 7.7: *Reconstruction using spline homotopy using same set of radial cross sections. Spline resonstruction clearly shows smoothness of the reconstruction.*

With the cubic homotopy it is possible to have a continuous reconstruction with no restriction on tangents at the beam ends, however the the resulting surface suffers from undesirable extremum points between places of high difference in radial distance between beam boundaries (see Figure 7.8). This is due to the fact that it is not always possible to have C^2 continuity while maintaining monotonicity [62, 144]. To overcome this problem, we introduce a shape preserving C^1 homotopy in the next subsection.

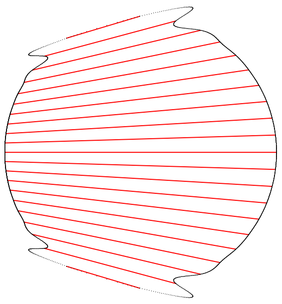


Figure 7.8: *Cubic spline reconstruction of a circle from a set of radial cross sections showing undesirable bulge on the sides.*

7.4.4 Shape preserving homotopy

Monotonicity preserving splines overcome the problem associated with cubic splines. Späth [131] introduced generalized exponential splines in tension that were further studied by Pruess [107] and others. These splines are piecewise exponential curves joining together to form a smooth curve in tension. The tension parameters, however, must be selected by some heuristic based on the gradient of the data points, or otherwise. Further, computation of tension splines is expensive due to evaluation of hyperbolic functions. The computation is also sensitive to the choice of tension parameters. This is specially true for very small tension parameters causing underflow of machine precision and for very large tension parameters causing overflow of machine precision [16].

Monotonicity can be attained by sacrificing smoothness while still using polynomials. We develop a C_1 homotopy based on the monotonic splines of Hymen [62]. This requires availability of derivatives $\partial\mathcal{H}_j(r,\lambda)/\partial\lambda$ at $\lambda = 0$. A monotone homotopy can be written in terms of Hermite basis functions as

$$\begin{aligned}\mathcal{H}_j(r,\lambda) =&(1-3\lambda^2+2\lambda^3)f_j(r)+(3\lambda^2-2\lambda^3)f_{j+1}(r)\\ &+(\lambda-2\lambda^2+\lambda^3)\left[\frac{\partial\mathcal{H}_j(r,\lambda)}{\partial\lambda}\right]_{\lambda=0}\\ &+(\lambda^3-\lambda^2)\left[\frac{\partial\mathcal{H}_{j+1}(r,\lambda)}{\partial\lambda}\right]_{\lambda=0}.\end{aligned} \tag{7.23}$$

The derivatives $\partial\mathcal{H}_j/\partial\lambda$ appearing in (7.23) enforce piecewise monotonicity in **c**. The de Boor and Swartz [37] piecewise monotonicity range can be extended for functions as

$$0 \leq \dot{f}_j \leq 3\min(\triangle f_j, \triangle f_{j+1}), \tag{7.24}$$

where, $\triangle f_j = (f_{j+1} - f_{j-1})$ and $\dot{f}_j$ denotes the required derivative. Starting with an approximation of the derivatives (either from a spline representation or differencing), these are then projected into the monotonicity region defined by (7.24) above according to

$$\dot{f}_j = \begin{cases}\min(\max(0,\dot{f}_j), 3\min(|\triangle f_j|, |\triangle f_{j+1}|))\\ \max(\min(0,\dot{f}_j), -3\min(|\triangle f_j|, |\triangle f_{j+1}|))\end{cases}. \tag{7.25}$$

The reader is referred to work by Hyman [62] for a detailed discussion on this. The constrained derivatives can be used in (7.23). The derivatives can only be computed numerically. Other procedures outlined by Paolo et al. [33] and Wolberg [144] employ optimizations to compute the derivatives. A two parameter family of monotone shape preserving homotopy can be formulated, as before, via tensor product. The result of monotone reconstruction is shown in section 7.6.

A reconstruction algorithm based on the developed homotopies is presented next.

7.5 Reconstruction algorithm

The MS70 sonar gives a 3D view of the ensonified volume as shown in Figure 7.1(b). The received signal represents the raw acoustic backscatter from the seabed and the fish school. In order to infer useful object information from the signals, these must first be corrected for spreading and absorption losses during acoustic wave propagation (see Simmonds and MacLennan [126] for details). The resulting *Volume backscattering strength* S_v values can then be used for analysis.

7.5.1 Characteristic function generation

To generate a good segmentation, anisotropic diffusion by Perona and Malik [106] may be performed on the Sv signals. The usual practice is to binary segment the signals by choosing a suitable threshold value depending on the species of the schooling fish seen in the volume. The threshold should be chosen so as to eliminate as much of the background and instrument noise as possible (see Simmonds and MacLennan [126]).

Following is the general procedure adopted to generate characteristic signals from raw acoustic signals.

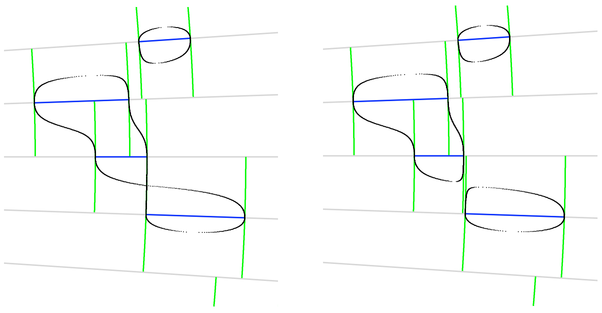
1. Perform anisotropic diffusion filtering on the input volume V_i to generate V_{diff}.
2. Generate a binary volume V_{bin} by using a threshold value S_{thres}.
3. Close holes in V_{bin} by 3D morphological closing (see Soille [129]). Isolated voxels are eliminated by a morphological opening.
4. Label the resulting volume, as V_{seg}, into different connected components (see Gonzalez and Woods [53]).
5. Select a component or combine several components from V_{seg} into V_χ that represents the object to reconstruct. This is a binary volume representing the characteristic signal of the desired object.

7.5.2 Solution of homotopy

Starting with a binary volume V_χ of size $M \times N \times R$, where R is the number of samples along a signal beam, we associate beam functions to all MN beams according to (7.6). A system of $(M-1) \times (N-1)$ homotopies are formulated.

The reconstruction $\mathcal{H} = 0$ traces the homotopy path without any problem except at places where the path encounters a singularity (self-intersection) in between the initial and terminal maps for $\lambda \in (0, 1)$. In such a case, the path becomes a non-manifold, as illustrated in Figure 7.9(a). Such a case can be avoided by first identifying a root r_k of any one of the beam functions that causes a singularity, and perturbing it such that the singularity in the path is avoided, as shown in Figure 7.9(b).

The set of homotopies can be solved to find points belonging to $\ker(\mathbf{H})$ that lie on the boundary of the reconstruction. An analytical solution to $\mathbf{H} = 0$ is not



(a) Singularity in the path (b) Avoiding singularity by perturbation of roots

Figure 7.9: *Handling cross-overs in the homotopy solution. Radial cross sections are shown in gray with location of roots along each radial line marked in green. A root on any radial line is projected on adjacent radial lines for clarity.*

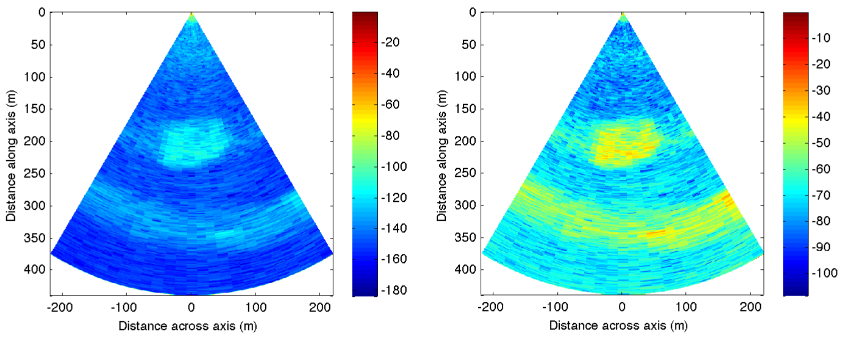
always possible, and a computational solution for all $\lambda \in [0, 1]$ is not realizable. Therefore, we construct a scalar field of $\mathbf{H}$ on a sampled grid. The homotopy path is then found by tracing zero level set of the scalar field.

7.6 Results

We present results of our reconstruction algorithm here. Although the reconstruction algorithm operates in the beam space, for sake of clarity we show all the 2D illustrations in polar coordinates. It must be noted that a coordinate transformation from beam space to spherical coordinates is needed only before the level set extraction and not before.

The raw data is shown in Figure 7.1(b) where the high intensity regions of the ensonified volume are rendered in red. These regions are mainly the fish school, seabed and noise at the top. We show a slice of the raw volume in Figure 7.10(a). Note that the intensity of the target diminishes as the distance from the transducer (located at the tip of the sector) increases. Figure 7.10(b) shows volume compensated for acoustic losses due to spreading and absorption. The intensities after compensation are distributed evenly (for example, typical fish school echo strength values are around -60 dB).

A threshold of -62 dB removes much of the background noise in this case. Com-



(a) Sector of the volume showing raw intensities (b) Volume corrected for spreading and absorption losses

Figure 7.10: *Intensity correction on raw volume.*

bined characteristic functions of the seabed and the fish school are shown in Figure 7.11. In the figure, thick black lines show the ranges where the characteristic function takes a value of 1.

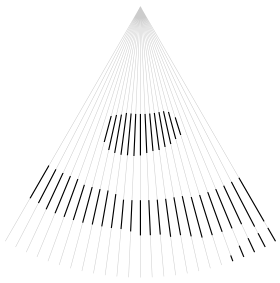
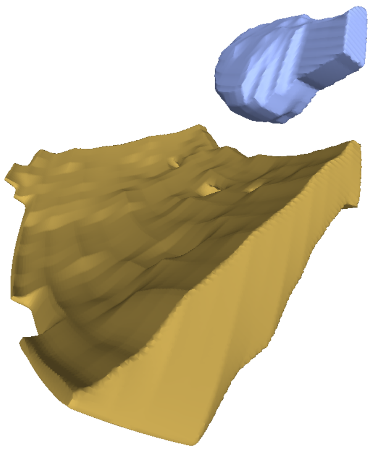
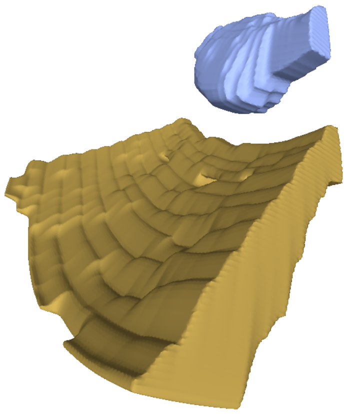
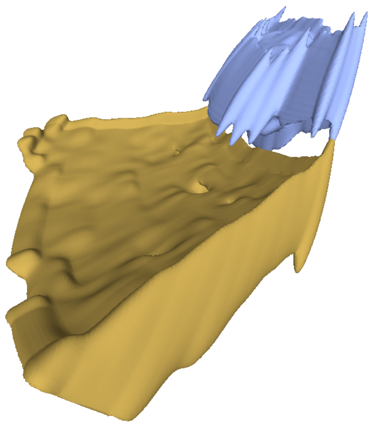


Figure 7.11: *Combined characteristic signal for seabed and fish school. Complete radial lines are shown in gray while the cross section is shown in black.*

A full reconstruction in $\mathbb{R}^3$ is shown in Figure 7.12. Here, different connected components are colored differently and the noise component is not considered during reconstruction. The seabed is shown in brown, while the fish school is colored in light blue. The difference in the reconstructions computed via the four homotopies is apparent and shows that the shape preserving monotone homotopy outperforms the others. A linear homotopy renders the reconstruction piecewise C^0 that does not look natural, while the non-linear homotopy introduces staircase like artifacts. The cubic homotopy on the other hand produces large variations in the surface at the sides causing undesired bumps.


(a) Linear homotopy


(b) Non-linear homotopy with $\eta = 2$


(c) Spline homotopy

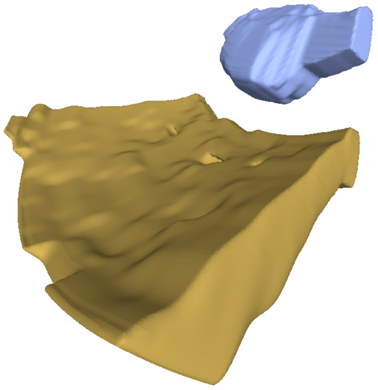

(d) Shape preserving homotopy

Figure 7.12: *Homotopy reconstruction of moving school of Sprat.*

With the real signals, it is not possible to quantify accuracy of the suggested homotopies. Therefore, we perform simulation tests described next.

7.6.1 Accuracy comparison

In order to evaluate the relative performance of the suggested homotopies, we reconstruct various simple geometric primitives. We compare the reconstructed surfaces with the respective primitives by means of certain measures. In all the simulations, we use a set of 20×25 beams of length 100 units and spanning an angular volume of $45^\circ \times 60^\circ$ (see Figure 7.13). A primitive is placed symmetrically in the spanned volume such that the center of the primitive is halfway from the apex of the sonar. With the given beam density, it must be noted that any primitive is sampled sparsely (i.e., not following the Nyquist criterion), and therefore it is not possible to completely recover an object with all details. This represents the practical case with acoustic instruments where beam density is constrained by various physical factors and cannot be arbitrarily increased. Also, beneath the water, objects to be mapped have unknown geometry and location. In our analysis, we use several parameters for shape comparison. Two simple parameters are ratio of the two surface areas A_r/A_o and ratio of the two volumes V_r/V_o. Further, we measure the reconstruction error by means of the *symmetrical Hausdorff distance* which is a good measure of the distance between two manifolds (see Aspert et al. [6]).

Symmetrical Hausdorff distance, d_H, between two surfaces M_0 and M_1 is given by

$$d_H(M_0, M_1) = \max\left\{ \sup_{x_0 \in M_0} \inf_{x_1 \in M_1} d(x_0, x_1), \right. \\ \left. \sup_{x_1 \in M_1} \inf_{x_0 \in M_0} d(x_0, x_1) \right\}, \tag{7.26}$$

where $d(\cdot, \cdot)$ is an appropriate metric for measuring distance between two points in a metric space. $d_H(M_0, M_1)$ measures the maximum possible distance that will be required to travel from surface M_0 to M_1. We compute this metric and use it to quantify the error in reconstruction of a phantom surface. Similarly, a mean Hausdorff error $\overline{d_H}$ [6] can also be defined as

$$\overline{d_H}(M_0, M_1) = \max\left\{ \frac{1}{A_{M_0}} \iint_{x_0 \in M_0} d(x_0, M_1)\ \mathrm{d}M_0, \right. \\ \left. \frac{1}{A_{M_1}} \iint_{x_1 \in M_1} d(x_1, M_0)\ \mathrm{d}M_1 \right\}, \tag{7.27}$$

where A_{M_0} and A_{M_1} denote the area of M_0 and M_1 respectively and $\mathrm{d}M_0$ and $\mathrm{d}M_1$ denote a differential area element on M_0 and M_1 respectively. In the following tables, a % refers to the relative Hausdorff distance measured as the percentage of the model bounding box diagonal.

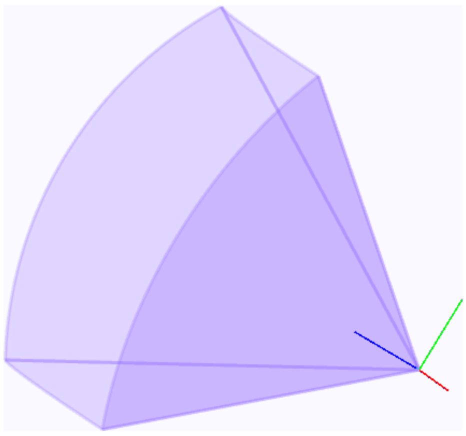
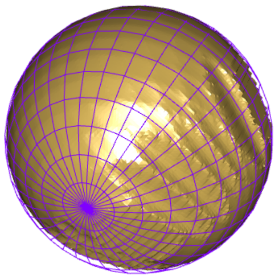
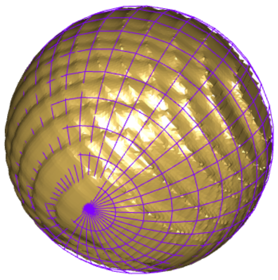


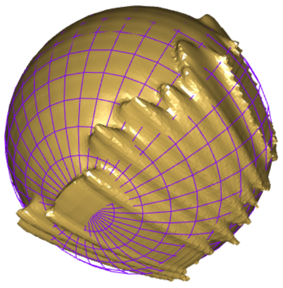
Figure 7.13: *The simulation test bed consisting of a set of simulated sonar beams* (20×25) *in a frustum of angular volume of* $45° \times 60°$. *The sonar beams emerge from the origin denoted by the intersection of the coordinate axes.*

The first primitive we consider is a sphere of radius 10 units. The reconstructions are shown in Figure 7.14 with the sphere drawn in wireframe. It can be seen that linear and monotone reconstructions produce satisfactory results, since both are shape preserving. Non-linear reconstruction produces the ringing effect while the cubic homotopy produces bumps in the surface at the sides. Table 7.1 shows the three shape measures for these reconstructions. The reconstructed volume and surface area for cubic reconstruction are larger than that of the sphere. The best ratios are obtained for the shape preserving C^1 reconstruction with an exception of the volume ratio for cubic reconstruction. The Hausdorff distances (both maximum and mean) are very high for the cubic reconstruction indicating that the reconstructed surface has regions of high deviation from the phantom sphere surface. The minimum error is obtained for the monotone reconstruction indicating that the two surfaces are closest to each other.

Next we consider a cube of side length 20 units. A cube is an interesting primitive since the surface is only C^0, therefore a linear reconstruction gives a shape that is closest to it (see Figure 7.15). The surface area ratio shown in Table 7.2 indicates that a better reconstruction is monotone. The volume ratios show that the reconstructed surface is an underestimation of the primitive except for the cubic reconstruction that has overhangs in the reconstruction. The Hausdorff error gives a clear indication that the cubic reconstruction deviates most from the phantom surface while the monotone reconstruction deviates the least.


(a) Linear homotopy


(b) Non-linear homotopy with $\eta = 2$


(c) Spline homotopy

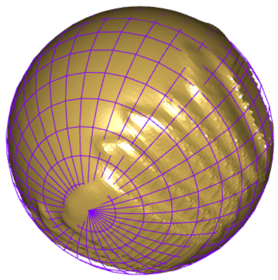
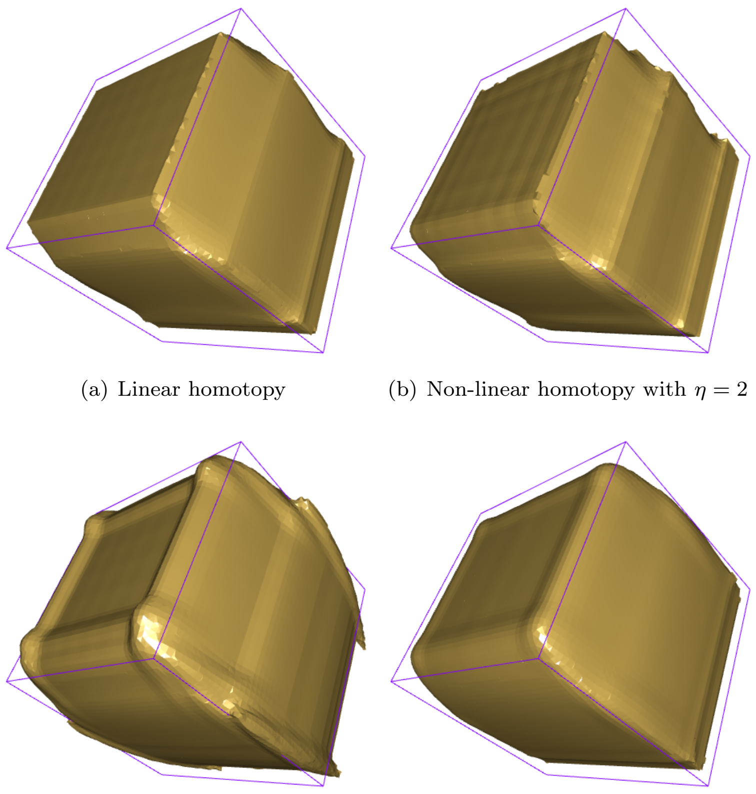

(d) Shape preserving homotopy

Figure 7.14: *Homotopy reconstruction of a sphere.*

Table 7.1: *Reconstruction performance with sphere*

Reconstruction	A_r/A_o	V_r/V_o	$d_H(\%)$	$\overline{d_H}(\%)$
Linear (C^0)	0.963	0.865	4.340	0.896
Non-linear $(C^{\eta-1})$	0.988	0.885	4.686	0.920
Cubic (C^2)	1.384	1.051	8.933	1.467
Monotone (C^1)	1.011	0.932	3.891	0.659



(a) Linear homotopy

(b) Non-linear homotopy with $\eta = 2$

(c) Spline homotopy

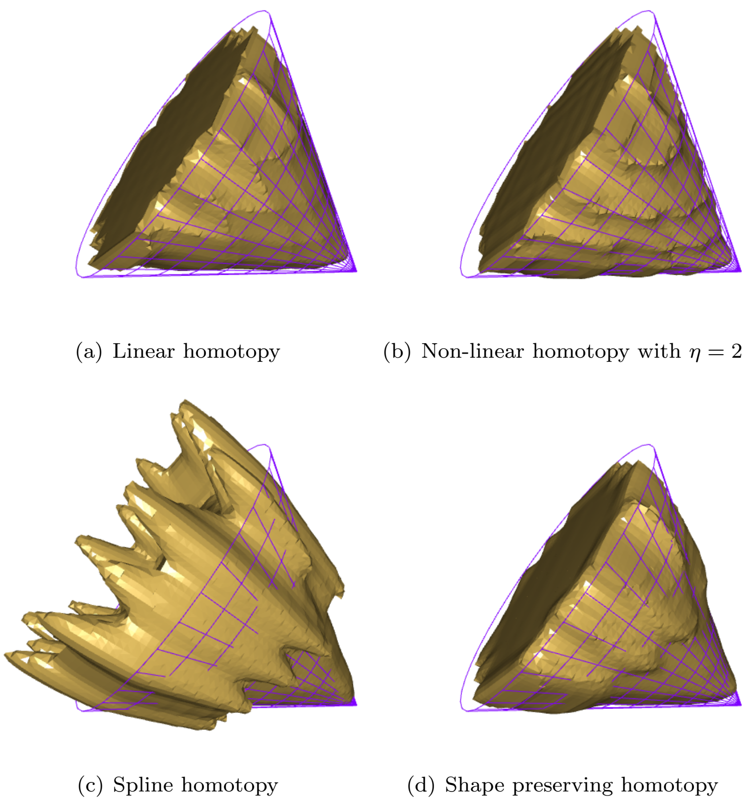
(d) Shape preserving homotopy

Figure 7.15: *Homotopy reconstruction of a cube.*

Table 7.2: *Reconstruction performance with cube*

Reconstruction	A_r/A_o	V_r/V_o	$d_H(\%)$	$\overline{d_H}(\%)$
Linear (C^0)	0.880	0.902	6.206	0.778
Non-linear($C^{\eta-1}$)	0.899	0.916	6.596	0.975
Cubic (C^2)	1.116	1.038	13.467	1.173
Monotone (C^1)	0.912	0.957	4.754	0.647

Next we experiment with a cone of semi-angle 30° and height 20 units, and a cylinder of radius 10 units and height 20 units. The reconstructions for these primitives are shown in Figs. 7.16 and 7.17, respectively. These surfaces have both planar and curved sections. As seen in the previous cases, the linear and monotone reconstructions perform better in this case as well with area and volume ratios closest to one, and least Hausdorff errors for the monotone reconstruction (as shown in Tables 7.3 and 7.5).



(a) Linear homotopy (b) Non-linear homotopy with $\eta = 2$

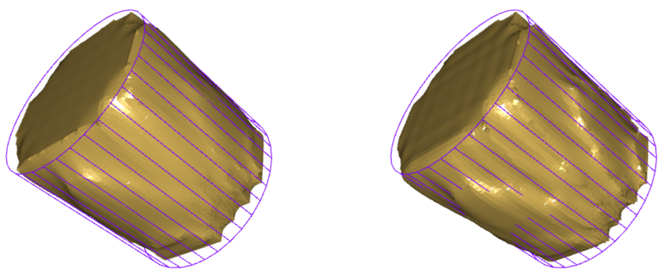
(c) Spline homotopy (d) Shape preserving homotopy

Figure 7.16: *Homotopy reconstruction of a cone.*

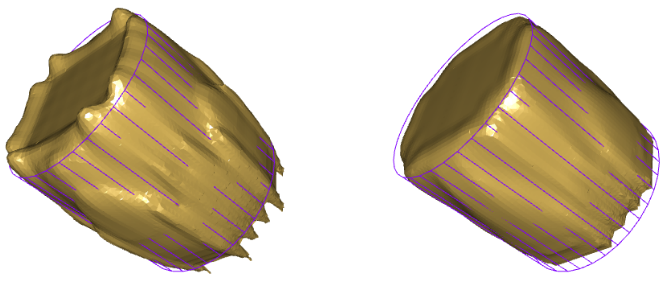
So far the primitives considered here are convex in geometry. Lastly, we consider a torus for reconstruction to show that a non-convex geometry with holes can be reconstructed in a similar fashion and poses no limitation on the algorithm developed here. Here again, the accuracy measures indicate that the monotone

Table 7.3: *Reconstruction performance with cone*

Reconstruction	A_r/A_o	V_r/V_o	$d_H(\%)$	$\overline{d_H}(\%)$
Linear (C^0)	0.859	0.858	8.012	0.950
Non-linear $(C^{\eta-1})$	0.885	0.876	7.640	0.998
Cubic (C^2)	1.460	1.186	19.049	2.509
Monotone (C^1)	0.901	0.944	6.979	0.641



(a) Linear homotopy (b) Non-linear homotopy with $\eta = 2$



(c) Spline homotopy (d) Shape preserving homotopy

Figure 7.17: *Reconstruction of a cylinder.*

Table 7.4: *Reconstruction performance with cylinder*

Reconstruction		A_r/A_o	V_r/V_o	$d_H(\%)$	$\overline{d_H}(\%)$
Linear	(C^0)	0.917	0.913	4.511	0.544
Non-linear	$(C^{\eta-1})$	0.944	0.926	4.862	0.731
Cubic	(C^2)	1.192	1.063	15.674	1.094
Monotone	(C^1)	0.954	0.975	4.104	0.377

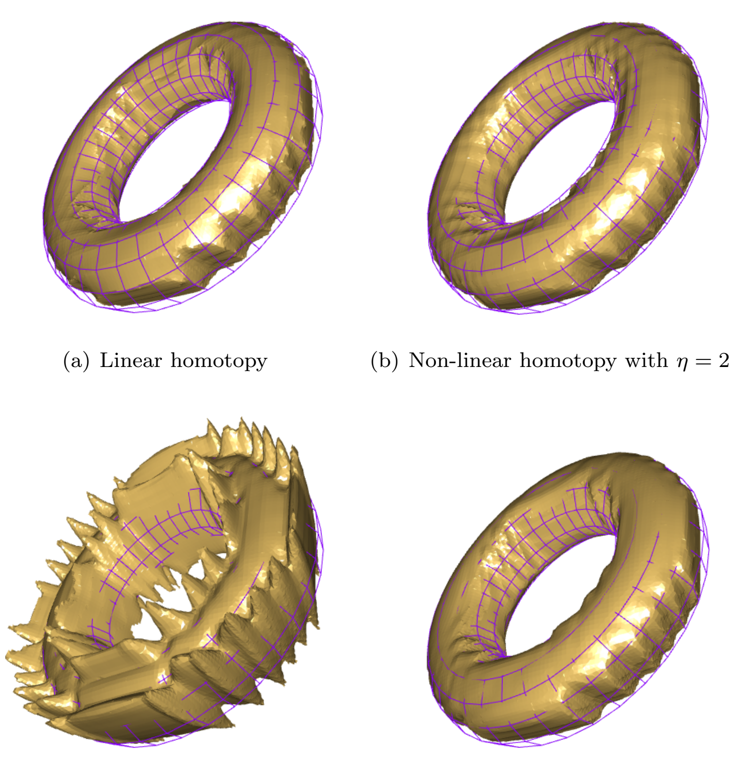
reconstruction is closest to the original torus model. Also looking at the results, the cubic reconstruction shows the presence of unwanted peaks.

Table 7.5: *Reconstruction performance with torus*

Reconstruction		A_r/A_o	V_r/V_o	$d_H(\%)$	$\overline{d_H}(\%)$
Linear	(C^0)	0.967	0.820	4.309	0.860
Non-linear	$(C^{\eta-1})$	0.984	0.842	4.514	0.782
Cubic	(C^2)	1.632	1.092	8.852	1.818
Monotone	(C^1)	1.000	0.889	3.847	0.625

The simulations shown here clearly indicate superiority of the monotone reconstruction compared to other homotopies. This is supported by the least Hausdorff errors in case of monotone homotopic reconstruction. The main advantage comes with suppression of bumps while still remaining smooth. The volume ratio does not seem to be the best indicator of the geometry of the reconstructed surface. Depending on the arrangement of intersecting beams, in some cases the deviation of this measure from one is less than such a deviation in other methods. This indicates that in those cases the cubic method performs better than other methods, which is clearly not the case as shown by the area ratio and the Hausdorff error metrics. If the geometry of the original object is known a priori, a suitable homotopy can be chosen. It must be noted that an increase in beam density will increase the accuracy of reconstruction.

To our knowledge, we presented an algorithm for direct reconstruction of a surface from one-dimensional cross sections embedded in 3D and the existing methods of surface reconstruction from acoustic cross sections rely on interpolation to first compute an acoustic image/volume before deriving a surface. Methods like the level set method operate on such a volume to reconstruct an


(a) Linear homotopy

(b) Non-linear homotopy with $\eta = 2$

(c) Spline homotopy

(d) Shape preserving homotopy

Figure 7.18: *Homotopy reconstruction of a torus.*

object using a deformable surface. A comparison of volume based methods with the proposed method will be not be justified due to different input types of these methods. In the next section we present the complexity of the presented algorithm.

7.7 Time complexity

The time complexity of the homotopy reconstruction algorithm depends on the number of linear cross sections $n_{sec} = MN$. The cost of assigning beam functions is $\mathcal{O}(n_r)$ per beam, where n_r is the number of roots along the beam. Since n_r depends on the object-beam intersection, it depends on the complexity of the objects considered. The complexity of formulating homotopies for the beams is $\mathcal{O}(n_{sec})$ for linear and non-linear homotopies. In case of spline homotopy, there is an additional one-time cost of inverting a matrix that amounts to a complexity of $\mathcal{O}((4(n_{sec}-1))^{2.376})$ using the Coppersmith-Winograd algorithm [32]. Such a computational cost can be drastically reduced by formulating a B-spline homotopy where every homotopy depends only on its four neighboring beam functions. Complexity of formulating the monotone homotopy is similar to the linear homotopy once the derivatives are computed (thereby suggesting the local nature). Computation of derivatives is an image space operation and therefore it is $\mathcal{O}(V)$, where V is the number of voxels in the grid of discretized space. Complexity of computing the solution path of the homotopy $\mathbf{H} = 0$ using isosurface extraction is $\mathcal{O}(V)$.

To get a notion of actual computation times for the reconstruction, a scene composed of two primitive objects (a cuboid and a sphere) is reconstructed from its intersection with sonar beams. The computational resource consisted of a 32-processor AMD Quad-Core machine with 256 GB of memory and an OpenMP implementation of the reconstruction algorithm running with 32 threads. Table 7.6 shows computation times in seconds for all the four reconstruction methods. Here, T_{bf} is the time taken in assigning beam functions to all the sonar beams, T_{inv} is the time taken in inverting the coefficient matrix (in case of cubic homotopy reconstruction), $T_{\mathcal{H}}$ is the time taken in evaluating the homotopy over a grid of volume, and T_{iso} is isosurface computation time. For monotone reconstruction method, $T_{\mathcal{H}}$ includes computation time of derivatives. The reconstruction is computed over a volume grid of size $154 \times 201 \times 201$.

Table 7.6: *Computation times (in seconds) for reconstruction. The scene composed of a cuboid and a sphere with the reconstruction performed on a raster volume grid of size* $154 \times 201 \times 201$.

	T_{bf}	T_{inv}	$T_{\mathcal{H}}$	T_{iso}
Linear	0.001328		1.071854	1.052210
Non-linear	0.001322		1.022518	1.307745
Cubic	0.001258	0.000247	4.100427	1.361817
Monotone	0.001273		3.801818	1.169319

7.8 Conclusions

We have developed a reconstruction algorithm based on homotopy continuation. Different formulations of homotopies suitable for smooth surface generation were presented. The resulting surface is of good quality both topologically and geometrically. The presented algorithm associates piecewise quadratic functions with the initial and terminal maps. In general, any smooth function that satisfies the design criteria can be used as a beam function. Furthermore, the results are readily extensible to higher dimensions. We conclude that homotopy based methods are quite powerful in predicting the information from the initial and the terminal maps.

Acknowledgement

The authors would like to thank collaborators from Simrad Kongsberg Maritime AS and DTU Aqua at the Technical University of Denmark. Special thanks to Lars Nonboe Andersen from Simrad and Bo Lundgren from DTU Aqua for fruitful discussions on Acoustic data processing.

Chapter 8

Homotopy based 2D topologic reconstruction from arbitrary linear cross sections

Ojaswa Sharma[†], **and François Anton**[†]

[†]Department of Informatics and Mathematical Modelling,
Technical University of Denmark, Lyngby, 2800, Denmark

Abstract

This paper addresses the problem of 2D object reconstruction from its 1D cross sections along straight lines. We impose no particular ordering restriction on the linear sections so that they can be arbitrary placed in the 2D plane. We propose a method based on homotopy continuation for topologically plausible reconstruction. By defining implicit functions on the computational domain, the topological properties of the object being reconstructed are preserved by the reconstruction.

8.1 Introduction

Object reconstruction from cross sections is a well known problem. Generally a spatial ordering within the cross sections aids reconstruction. We consider the case of reconstructing an object from arbitrary linear cross sections. Such cross sectional information can be obtained with many devices. An example is acoustic probes that obtain range information of the subject by sending an acoustic pulse.

The problem of reconstruction from arbitrary cross sections has been studied by Sidlesky et al. [125], Liu et al. [83], and Memari and Boissonnat [89]. Sidlesky et al.[125] define sampling conditions on the reconstruction, while in our reconstruction algorithm, we allow the sampling to be sparse. Our approach to reconstruction considers the "presence" or "absence" of information along any intersecting line. This is in contrast to work by Sidlesky et al. [125], where the authors consider that a line not intersecting the object does not contribute to the reconstruction. In our algorithm, such a line is considered to contribute to the reconstruction by defining a linear section, no part of which belongs to the reconstruction. Methods proposed by Liu at al. [83], and Memari and Boissonnat [89] are both based on Voronoi diagrams. Memari and Boissonnat also provide sampling conditions for topological correctness of their reconstruction. However, their reconstruction consists of line segments in the plane that originate from the Delaunay complex of the cross sections, thus resulting in a C^0 reconstruction curve in the plane. On the contrary, our reconstruction is based on the solution of homotopy of polynomials resulting in smoothness that depends on the order of the chosen polynomial.

This paper is organized as follows. Section 8.2 defines the reconstruction problem mathematically. We introduce the concept of homotopy continuation in section 8.3 followed by the main reconstruction algorithm in section 8.4. In the same section 8.4, we discuss our edge barycentric coordinates on convex polygons and provide details of our homotopy based reconstruction algorithm. Finally we provide results of the reconstruction and accuracy using hierarchical sampling.

8.2 Problem definition

Given a set of lines $\{\mathcal{L}_i : i \in [0, n-1]\}$ in a plane, intersecting an object $\mathcal{O}$ along segments $\{\mathcal{S}_{i,j} : j \in [0, m_i - 1]\}$, the problem of object reconstruction from arbitrary linear cross sections is to reconstruct an object $\mathcal{O}$ from $\mathcal{S}_{i,j}$ such that the reconstruction $\mathcal{R}$ satisfies

$$\mathcal{L}_i \bigcap \mathcal{O} = \mathcal{L}_i \bigcap \mathcal{R}, \tag{8.1}$$

and that $\mathcal{R}$ is homeomorphic to $\mathcal{O}$. Further, the reconstruction should also be geometrically close to the object. We quantify the geometric closeness in our reconstruction by means of several area based ratios viz. the ratio of area of reconstruction and the area of the object, and the ratio of the absolute difference of the two areas and the area of the object. Length ratio is also a good indicator of geometric closeness. Furthermore, Hausdorff distance between the two curves gives a good measure of the distance between them.

In this context we impose no restrictions on the ordering or arrangement of the intersecting lines. However, the placement of intersecting lines plays an important role in the correctness of the reconstruction. A placement that covers salient object features results in a better reconstruction. In order to quantify an optimal placement, consider a set of intersecting lines in a plane along with the object to be reconstructed. The intersecting lines partition the object into smaller regions. Considering the simply connected boundary of the object that belongs to a region (see the highlighted curve segment in Figure 8.1(a)), *tortuousity* [42], which gives a simple measure of how twisted a curve is, can be computed. It is defined as the ratio of curve length and the span of the curve $y = f(x)$ in one direction

$$\tau(f) = \frac{\int_a^b \sqrt{1 + \left(\frac{df}{dx}\right)^2}}{b - a} = \frac{L}{C}, \tag{8.2}$$

where a and b are real numbers, L is the length of the curve, and C is the span of curve. τ can be considered as a measure of straightness of a curve. For parts of the object boundary that are not intersected by the lines, we define τ to be ∞.

It is not difficult to see that higher the value of τ for any region, more it is susceptible to generate part of the reconstruction that is non-homeomorphic to the object. However, if the sampling is such that the intersecting lines are chosen along the medial axis of the object, then such regions can be avoided (see Figure 8.1(b)). In that case, τ remains close to one for different regions. Such a sampling is illustrated in subsection 8.5.1 for deriving accuracy statistics for the proposed reconstruction method. The endoskeleton and the exoskeleton provide optimal placement of the cutting lines along the loci of high curvature of the object boundary.

Another desired trait of a reconstruction is smoothness, and we will show that

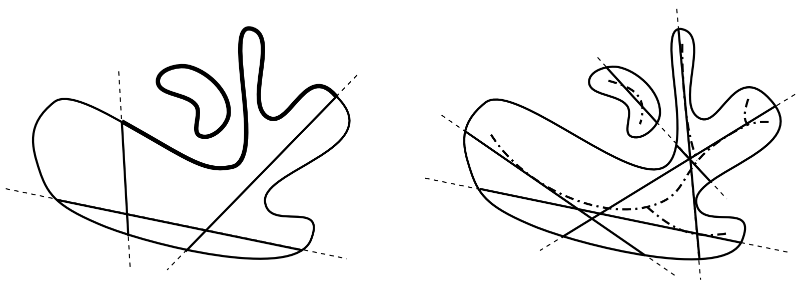

(a) Undersampling (b) Optimal sampling along medial axis

Figure 8.1: *Sampling condition on the intersecting lines.*

the proposed method of continuous deformations results in a reconstruction that is at least C^1.

8.3 Homotopy continuation

Homotopy is, roughly speaking, concerned with identification of paths between objects that can be continuously deformed into each other. The history of study of homotopy dates back in the late 1920's when the the homotopy theory was formalized [66].

Definition 8.1 Let $f : X \mapsto Y$ and $g : X \mapsto Y$ be two continuous maps between topological spaces X and Y. These maps are called *homotopic*, $f \simeq g$, if there is a *homotopy* or a continuous map $\mathcal{H} : X \times [0,1] \mapsto Y$ between them, such that $\mathcal{H}(x,0) = f(x)$ and $\mathcal{H}(x,1) = g(x)$ for all $x \in X$.

Therefore, we can write the homotopy $\mathcal{H}_\lambda : X \mapsto Y$ as

$$\mathcal{H}_\lambda(x) = \mathcal{H}(x, \lambda), \tag{8.3}$$

and thus, $\mathcal{H}_0 = f$ and $\mathcal{H}_1 = g$. One can visualize how the deformation $\mathcal{H}_\lambda$ continuously takes f to g (see Figure 8.2) by varying the parameter λ.

One can impose additional constraints on the deformation path. For example, a specific constraint on fixed endpoints leads to homotopy of paths. For two pairs of homotopic maps $X \xrightarrow[f\simeq g]{} Y \xrightarrow[\bar{f}\simeq \bar{g}]{} Z$, the compositions $\bar{f} \circ f$ and $\bar{g} \circ g$ are also

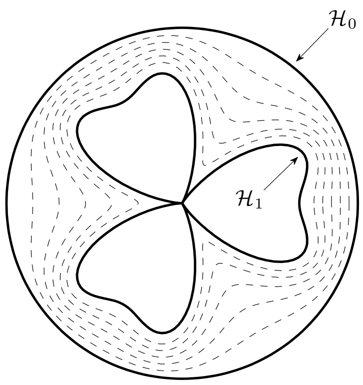


Figure 8.2: *Continuous deformation of a circle into Habenicht's clover.*

homotopic via the composition $\bar{\mathcal{H}}_\lambda \circ \mathcal{H}_\lambda$. Further, for two pairs of homotopic maps $f_i \simeq g_i : X_i \mapsto Y_i, i = 1, 2$, the maps $f_1 \times f_2$ and $g_1 \times g_2$ from $X_1 \times X_2$ into $Y_1 \times Y_2$ are also homotopic via $\mathcal{H}_\lambda^{(1)} \times \mathcal{H}_\lambda^{(2)}$, in which case it is called a *product homotopy* [66].

Continuous deformations have been successfully used to solve non-linear system of equations that are otherwise hard to solve. A homotopy tries to solve a difficult problem with unknown solution by starting with a simple problem with known solution. Stable predictor-corrector and piecewise-linear methods for solving such problems exist (see Allgower and Georg [3]). The system $\mathcal{H}(x, \lambda) = 0$ implicitly defines a curve or 1-manifold of solution points.

Given a smooth $\mathcal{H}$ and an $u_0 \in \mathbb{R}^{N+1}$ such that $\mathcal{H}(u_0) = 0$ and $\text{rank}(\mathcal{H}'(u_0)) = N$, there exists a smooth curve $c : \alpha \in J \mapsto c(\alpha) \in \mathbb{R}^{N+1}$ for some open interval J containing zero such that for all $\alpha \in J$ (Allgower and Georg [3])

1. $c(0) = u_0$,
2. $\mathcal{H}(c(\alpha)) = 0$,
3. $\text{rank}(\mathcal{H}'(c(\alpha))) = N$,
4. $c'(\alpha) \neq 0$.

In this work, we use homotopy or continuous deformations for object reconstruction. This is discussed in the next section.

8.4 Reconstruction algorithm

We start with a set of lines $\{\mathcal{L}_i\}$ and cross sections $\mathcal{S} = \{\mathcal{S}_{i,j} : \mathcal{S}_{i,j}$ is the j^{th} cross section on $\mathcal{L}_i\}$ in a plane, with the reconstruction domain restricted to the extended bounding box $\mathcal{B}_{box}$ of the cross sections. The set of lines $\{\mathcal{L}_i\}$ partition $\mathcal{B}_{box}$ into a set of convex polygons $\{\mathcal{G}_k : k \in [0, p-1]\}$. This is shown in Figure 8.3 where $\mathcal{O}$ is drawn dotted, the intersecting lines are shown dashed with the cross sections as thick solid lines, and the boundary of the extended bounding box shown dashed. Our reconstruction algorithm consists of assigning a homotopy $\mathcal{H}_k$ to every $\mathcal{G}_k$. The reconstruction is then obtained as

$$\mathcal{R} = \bigcup_k \{(x, y) : \mathcal{H}_k = 0\}. \tag{8.4}$$

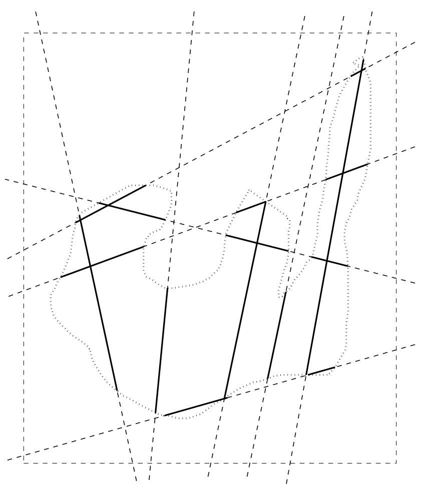


Figure 8.3: *A set of lines intersecting an object (dotted).*

A homotopy can be seen as a smooth transition from one map to another. We can extend this definition to multiple maps by defining a homotopy in multiple variables

$$\mathcal{H}(\lambda_0, \lambda_1, \cdots, \lambda_{s-1}) = \sum_{t=0}^{s-1} f_t \lambda_t, \tag{8.5}$$

with

$$\sum_{t=0}^{s-1} \lambda_t = 1, \ \lambda_t \geq 0. \tag{8.6}$$

8.4.1 Edge maps

Using (8.5), a smooth map can be defined over $\mathcal{G}_k$ for a choice of maps $\{f_t : t \in [0, s_k - 1]\}$ defined on s_k edges of $\mathcal{G}_k$. Let these maps be called *edge maps.* Since a homotopy is defined over each polygon, continuity in $\mathcal{H}$ at the polygon boundaries is required. For continuity across all polygons, the definition of the edge maps must be consistent. Since polygon edges are a subset of the cross section lines, it suffices to define edge maps over $\{\mathcal{L}_i\}$.

An edge map f_i should completely describe the boundary, interior and exterior of the intersection of $\mathcal{L}_i$ with $\mathcal{O}$. To define f_i, we associate a local coordinate system with each line with the x-axis measuring distance r along $\mathcal{L}_i$ from a chosen origin. Given abscissae $r_q, q \in [0, 2j-1]$ of the intersections $\mathcal{S}_{i,j}$ on $\mathcal{L}_i$, we define the corresponding edge map as a C^1 piecewise quadratic polynomial

$$f_i(r) = \sum_{q=0}^{2m_i-2} \frac{\alpha_q(-r^2 + r(r_q + r_{q+1}) - r_q r_{q+1}))}{(r_{q+1} - r_q)}, \tag{8.7}$$

where α_q is the positive gradient $\left|\frac{\mathrm{d}\,f}{\mathrm{d}\,r}\right|_{r=r_q}$ defined as

$$\alpha_q = (-1)^{q+1}\alpha_0, \tag{8.8}$$

with α_0 being a chosen positive slope at r_0. Figure 8.4 illustrates such an edge map. Note that an edge map f_i should not be interpreted geometrically in $\mathbb{R}^2$, but in $\mathbb{R}^3$.

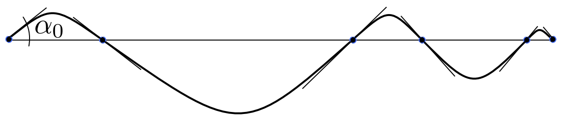


Figure 8.4: *Piecewise quadratic function as an edge map.*

8.4.2 Barycentric coordinates

It is natural to consider barycentric coordinates of a polygon as homotopy variables because of the two useful properties that they offer. Barycentric coordinates span a complete polygon and are a partition of unity. Traditional barycentric coordinates for a point in a triangle (and simplices in general) are defined by its vertices. Relevant generalization of barycentric coordinates to polygons were provided by Wachspress [138] and later by Meyer et al. [90]. In the current context, we define barycentric coordinates in terms of the edges of a polygon rather than the vertices. Such a definition allows us to apply the concepts developed so far to associate a suitable homotopy to a polygon.

8.4.2.1 Barycentric Coordinates based on Orthogonal Distance

Consider a polygon $\mathcal{G}_1$ and one of its sides e_i with end points p_i and p_{i+1}. For any point p inside $\mathcal{G}_1$, let $\hat{p}$ be the foot of the perpendicular from p on e_i. Let d_i be the distance from the local origin O_j of line $\mathcal{L}_j$ (corresponding to edge e_i) and $\hat{p}$, and h_i be the distance between p and $\hat{p}$ (see Figure 8.5). We suggest the following barycentric coordinates on the edges of polygon $\mathcal{G}_1$

$$\lambda_i = \frac{1/\psi(h_i)}{\displaystyle\sum_{j=0}^{s-1} 1/\psi(h_j)}, i \in [0, s-1], \tag{8.9}$$

where $\psi : \mathbb{R} \mapsto \mathbb{R}$ is a monotonically increasing smooth function with $\psi(0) = 0$. Defined in this way, the barycentric coordinates satisfy positivity, partition of unity, and smoothness. It can be shown that for any edge e_k, when $h_k \to 0$, we get the following limits: $\lambda_k \to 1$ and $\lambda_{i,i\neq k} \to 0$.

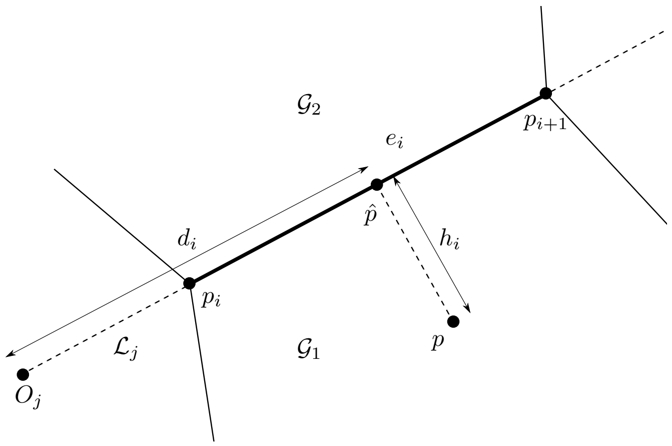


Figure 8.5: *Setup for defining barycentric coordinates.*

Barycentric coordinates defined over the edges of the polygon allow us to formulate a smooth deformation of the edge maps of the polygon over the polygon.

8.4.3 Homotopy

Equipped with the above defined edge maps and barycentric coordinates for any polygon $\mathcal{G}_k$, we define a homotopy $\mathcal{H}_k$ as

$$\mathcal{H}_k(p) = \sum_{i=0}^{s_k-1} f_i\left(d_i(p)\right)\lambda_i(p), \tag{8.10}$$

where $d_i(p)$ is the distance along line $\mathcal{L}_j$ as discussed in (8.4.2.1). The homotopy (8.10) continuously deforms edge maps f_i within the polygon and thus generates a smooth field.

Across polygons, the homotopies are continuous and at least C^1 smooth (see Appendix A for the proof). However, at all the intersection points of the lines with the boundary $\partial\mathcal{O}$ of the object $\mathcal{O}$, $\mathcal{Q} = \{\mathcal{Q}_{i,j} : \mathcal{Q}_{i,j} \in \mathcal{L}_i \bigcap \partial\mathcal{O}\}$, the generated curve $\mathcal{H}^{-1}(0)$ is orthogonal to lines $\mathcal{L}_i$ (see Appendix A). Therefore, the resulting reconstruction is somewhat unnatural. Given normals at the intersection points $\mathcal{Q}$ (which is the case with many range scanning physical devices), we propose a tangent alignment scheme for the resulting curve by locally warping the domain of the homotopies.

8.4.4 Tangent alignment using local space rotations

Given unit normals $\widehat{\mathcal{N}}$ at intersection points $\mathcal{Q}$, the reconstruction can be constrained to be normally aligned to these normals at these points. We enforce this constraint by local space rotations around points $\mathcal{Q}$.

The reconstruction $H^{-1}(0)$ is orthogonal to the intersecting lines at $\mathcal{Q}$. Starting with a point p in the neighborhood of one of the points p_c of $\mathcal{Q}_{k,j}$ lying on $\mathcal{L}_k$, we rotate p about p_c by an angle $-\theta$ to give point $\widetilde{p}$ in the plane (see Figure 8.6). The angle θ is chosen to be the signed angle between $\widehat{\mathcal{N}}$ and $\nabla\mathcal{L}_k$ about the point p_c. The resulting homotopy $\widetilde{\mathcal{H}}$ for a polygon $\mathcal{G}$ can be written as

$$\widetilde{\mathcal{H}} = \sum_{i=0}^{s_k-1} f_i\left(\widetilde{d}_i\right)\widetilde{\lambda}_i, \tag{8.11}$$

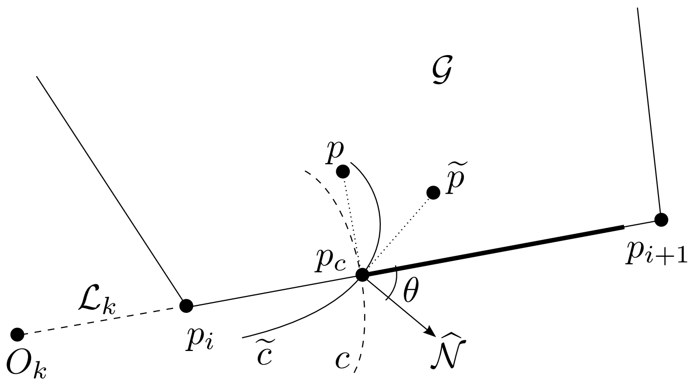


Figure 8.6: *Rotation of a point for tangent alignment.*

where,

$$\widetilde{d}_i = ||O_k - p_i|| + \frac{\triangle x_i(\widetilde{x} - x_i) + \triangle y_i(\widetilde{y} - y_i)}{l_i}, \tag{8.12}$$

$$\widetilde{\lambda}_i = \frac{1/\psi(\widetilde{h}_i)}{\sum_{j=0}^{s-1} 1/\psi(\widetilde{h}_j)}, i \in [0, s-1], \text{ and} \tag{8.13}$$

$$\widetilde{h}_i = \frac{\triangle y_i(\widetilde{x} - x_i) - \triangle x_i(\widetilde{y} - y_i)}{l_i}. \tag{8.14}$$

The rotated point $\widetilde{p}$ can be written as

$$\begin{aligned}\widetilde{p} = \begin{pmatrix} \widetilde{x} \\ \widetilde{y} \end{pmatrix} &= \begin{pmatrix} x_c \\ y_c \end{pmatrix} + \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} \\ &= p_c + \mathbf{R}_{-\theta}(p - p_c) \end{aligned} \tag{8.15}$$

The modified gradient of the homotopy in the neighborhood of p_c can be now computed using the chain rule. The partial derivative of $\widetilde{\mathcal{H}}$ w.r.t. x is

$$\begin{aligned}\frac{\partial \widetilde{\mathcal{H}}}{\partial x} = \sum_{i=0}^{s-1} \Bigg(& f_i'(\widetilde{d}_i) \left[\frac{\partial \widetilde{d}_i}{\partial \widetilde{x}} \frac{\partial \widetilde{x}}{\partial x} + \frac{\partial d_i}{\partial \widetilde{y}} \frac{\partial \widetilde{y}}{\partial x} \right] \widetilde{\lambda}_i \\ & + f_i(d_i) \left[\frac{\partial \widetilde{\lambda}_i}{\partial \widetilde{x}} \frac{\partial \widetilde{x}}{\partial y} + \frac{\partial \widetilde{\lambda}_i}{\partial \widetilde{y}} \frac{\partial \widetilde{y}}{\partial y} \right] \Bigg) \end{aligned} \tag{8.16}$$

In the limit as point $h_k \to 0$ (or $p \to p_c$), using a similar derivation as in Appendix A, we can write

$$\begin{aligned}
\lim_{h_k \to 0} \frac{\partial \widetilde{\mathcal{H}}}{\partial x} =& f'_k(\widetilde{d}_k) \left[\frac{\partial \widetilde{d}_k}{\partial \widetilde{x}} \frac{\partial \widetilde{x}}{\partial x} + \frac{\partial d_k}{\partial \widetilde{y}} \frac{\partial \widetilde{y}}{\partial x} \right] \\
=& f'_k(\widetilde{d}_k) \left[\frac{\triangle x_k}{l_i} \cos\theta - \frac{\triangle y_k}{l_i} \sin\theta \right], \text{ and} \\
\lim_{h_k \to 0} \frac{\partial \widetilde{\mathcal{H}}}{\partial y} =& f'_k(\widetilde{d}_k) \left[\frac{\triangle x_k}{l_i} \sin\theta + \frac{\triangle y_k}{l_i} \cos\theta \right]. \qquad (8.17)
\end{aligned}$$

We know that the gradient

$$\lim_{h_k \to 0} \nabla \mathcal{H} = \left(f'_k(d_k) \frac{\triangle x_k}{l_k}, f'_k(d_k) \frac{\triangle y_k}{l_k} \right). \qquad (8.18)$$

From (8.18) and (8.17) it can be seen that in the limit $h_k \to 0$

$$\frac{\nabla \widetilde{\mathcal{H}}}{||\nabla \widetilde{\mathcal{H}}||} = \mathbf{R}_\theta \left(\frac{\nabla \mathcal{H}}{||\nabla \mathcal{H}||} \right). \qquad (8.19)$$

Therefore, we can achieve the desired rotation of the reconstruction curve by rotating the local coordinates around points $\mathcal{Q}$ in the opposite direction.

8.4.4.1 Smooth rotations of the reconstruction curve

In order to generate a smooth distortion $\widetilde{H}^{-1}(0)$ of the curve $H^{-1}(0)$ the neighborhoods of points $\mathcal{Q}$ must be carefully chosen. A natural neighborhood for points in $\mathcal{Q}$ is their respective Voronoi polygons. However, a constant rotation for all the points in a particular Voronoi region results in a discontinuous curve at the boundary of these polygons. Therefore, we seek a continuous weight function w_{p_i} inside the Voronoi region $\mathcal{V}_{\mathcal{Q}}(p_i)$ of any generator $p_i \in \mathcal{Q}$ such that

$$\begin{aligned}
&w_{p_i}(p_i) = 1, \text{ and} \\
&w_{p_i}(\partial \mathcal{V}_{\mathcal{Q}}(p_i)) = 0. \qquad (8.20)
\end{aligned}$$

These requirements on the weight function impose a smooth transition of rotations from one influence zone to another and ensure little or no rotation at points far away from the centers of rotations. For any point p inside $\mathcal{V}_{\mathcal{Q}}(p_i)$, consider its nearest neighbor p_i and the next nearest neighbor $p_i^{(2)} \in \mathcal{Q}$. Denote by $d_1(p)$ the distance between p and p_i and by $d_2(p)$ the one between p and $p_i^{(2)}$. We can formulate the required weight function as

$$w_{p_i}(p) = \frac{d_2(p) - d_1(p)}{d_2(p) + d_1(p)}. \qquad (8.21)$$

The first nearest neighbor p_i is the generator point p_i of $\mathcal{V}_{\mathcal{Q}}(p_i)$. The second nearest neighbor $p_i^{(2)}$ can be found by computing the second order Voronoi diagram of $\mathcal{Q}$. The weight function resulting from (8.21) is shown in Figure 8.7. The complete method is outlined in Algorithm 8.1.

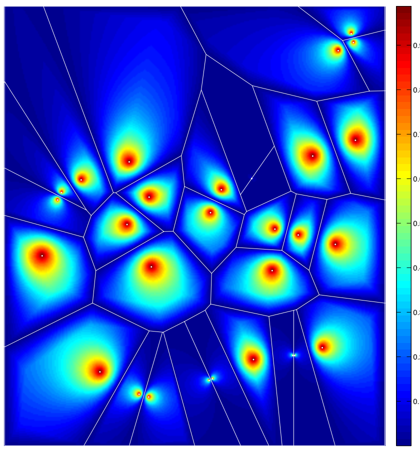


Figure 8.7: *Weights based on higher order Voronoi diagram.*

8.5 Results

We test our reconstruction method on a hand drawn curve intersected by a set of arbitrarily oriented lines. The lines yield a polygonal tessellation in the plane. We reconstruct the original curve from the intersections over the polygons of this tessellation.

The choice of ψ influences the overall shape of the curve $H^{-1}(0)$. This curve is guaranteed to pass through the end-points of the intersections. Figures 8.8(a), 8.9(a), 8.10(a), and 8.11(a) shows results of our reconstruction for four choices of the function. The proof given in Appendix A shows that the reconstructed curve is always orthogonal to the intersecting lines. For $\psi(x) = x$, the orthogonality is not apparent in Figure 8.8(a) at a large scale. We note that for a flatter curve ψ, the reconstruction becomes orthogonal to the line segments earlier while approaching them. This can be seen for $\psi(x) = 2x^4 + 3x^6$ in Figure 8.10(a).

We also show results of the reconstructed curve after applying local rotations at

```
Input: $\{\mathcal{S}_{i,j}\}$ on $\{\mathcal{L}_i\}$, $\widehat{\mathcal{N}}$ at $\mathcal{Q}$
Output: $\mathcal{R}$
Define an edge map $f_i$ on each $\mathcal{L}_i$ using $\{\mathcal{S}_{i,j}\}$
Partition $\mathcal{B}_{box}$ into polygonal tiles $\{\mathcal{G}_k\}_{k=0}^{k<p}$ with $\{\mathcal{L}_i\}$
Compute first order Voronoi diagram $\mathcal{V}_q^1$ of $\mathcal{Q}$
Compute second order Voronoi diagram $\mathcal{V}_q^2$ of $\mathcal{Q}$
Compute $w$ from $\mathcal{V}_\mathcal{Q}^1$ and $\mathcal{V}_\mathcal{Q}^2$
for $k \in [0, p-1]$ do
    Compute Voronoi diagram $\mathcal{V}_k$ of $\mathcal{G}_k$
    for $\mathbf{x} \in \mathcal{G}_k$ do
        Let $\mathbf{x}_c \in \mathcal{Q}$ be the generator of Voronoi polygon of $\mathbf{x}$
        $\widetilde{\mathbf{x}} \leftarrow \mathbf{x}_c + \mathbf{R}(-\theta)(\mathbf{x} - \mathbf{x}_c)$
        Compute $\{\lambda_i(\widetilde{\mathbf{x}})\}_{i=0}^{i<s}$ for all $s$ edges of $\mathcal{G}_k$
        Compute $d_i(\widetilde{\mathbf{x}})$ for all edges of $\mathcal{G}_k$
        Compose $\widetilde{\mathcal{H}}_k$
    end
end
$\widetilde{\mathcal{H}} \leftarrow \bigcup \widetilde{\mathcal{H}}_k$
$\mathcal{R} \leftarrow \ker \widetilde{\mathcal{H}}$
```

Algorithm 8.1: The reconstruction algorithm.

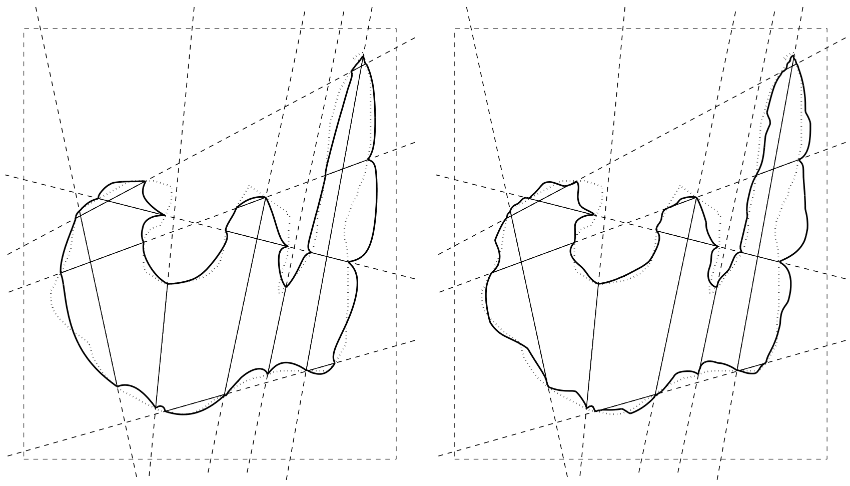
points $\mathcal{Q}$. The local rotations distort the original curve in the desired direction as seen in Figures 8.8(b), 8.9(b), 8.10(b), and 8.11(b).

The next section discusses accuracy statistics of our method based on how much information is provided to the reconstruction algorithm in terms of the number of intersections.

8.5.1 Reconstruction accuracy

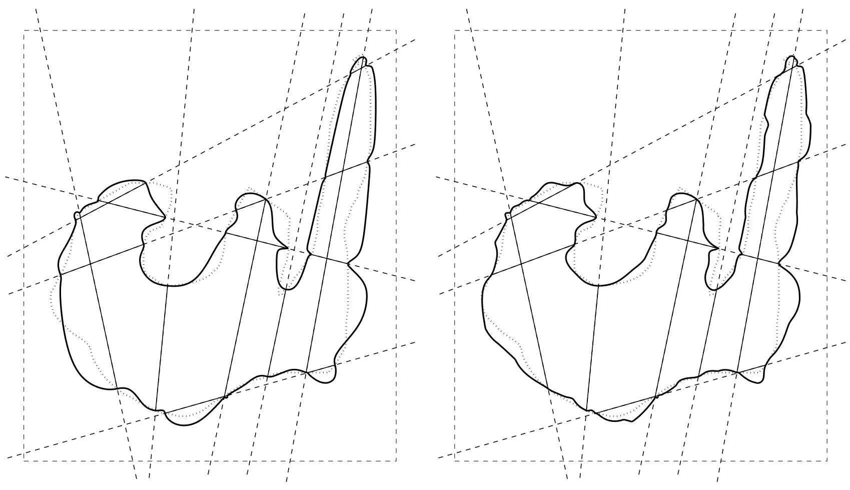
A good reconstruction depends on the choice of cutting lines placed carefully to cover salient geometric features of the object to be reconstructed. In order to test the reconstruction algorithm for accuracy, we sample intersecting lines from the skeleton of the test object. We choose to sample the skeleton of the object since it captures the salient details of the object. To show the dependence of reconstruction accuracy on the number of intersecting lines, a hierarchy of skeletons is used.

A skeleton hierarchy is computed from a straight line skeleton [2]. The hierarchy



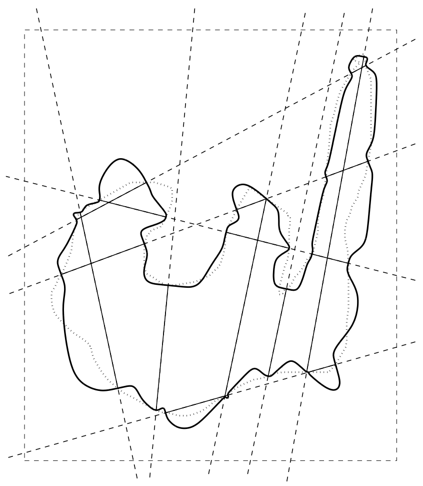
(a) $\mathcal{H}^{-1}(0)$ (b) $\widetilde{\mathcal{H}}^{-1}(0)$

Figure 8.8: *Reconstruction with $\psi(x) = x$.*



(a) $\mathcal{H}^{-1}(0)$ (b) $\widetilde{\mathcal{H}}^{-1}(0)$

Figure 8.9: *Reconstruction with $\psi(x) = x/L$.*


(a) $\mathcal{H}^{-1}(0)$

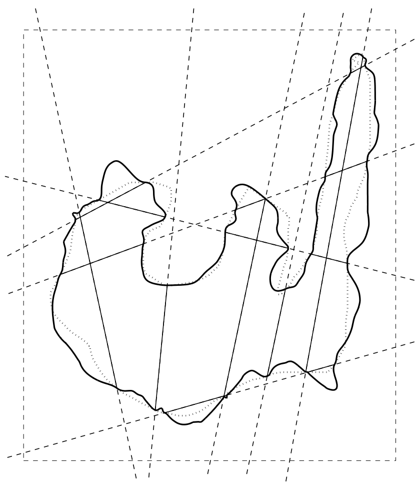
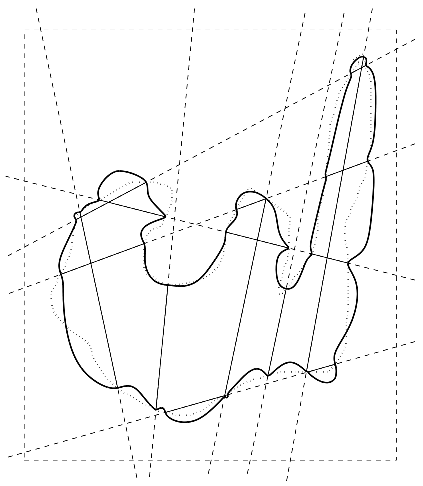

(b) $\widetilde{\mathcal{H}}^{-1}(0)$

Figure 8.10: *Reconstruction with* $\psi(x) = 2x^4 + 3x^6$.


(a) $\mathcal{H}^{-1}(0)$

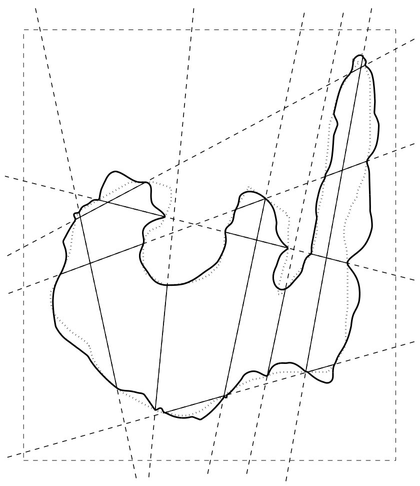

(b) $\widetilde{\mathcal{H}}^{-1}(0)$

Figure 8.11: *Reconstruction with* $\psi(x) = -(e^{-1})^{e^{-1}} + (x + e^{-1})^{(x+e^{-1})}$.

provides an incremental simplification of the skeleton based on discrete curve evolution of the skeleton branches [77, 8]. Curve simplification by discrete curve evolution uses local angles at every vertex of the curve to assign a relevance to each vertex. Based on the computed relevance values, the curve is evolved (simplified) by deletion of vertices. Figure 8.12 shows such a skeleton hierarchy with five levels. The base skeleton is computed using the straight line skeleton module of the CGAL library [24] .

Sampled segments from the skeleton at every level are used as cutting lines for reconstruction. Thus, for every level of hierarchy of skeletons, we compute a reconstruction of the object. We next compute various error metrics for different reconstructions thus obtained. To get a comprehensive idea of the reconstruction accuracy, metrics based on area, mean distance error, and curve lengths are considered here.

Denoting the area of the model object by $\mathcal{A}_{mod}$, the area of the reconstruction at level i by $\mathcal{A}^i_{rec}$, the absolute difference of $\mathcal{A}_{mod}$ and $\mathcal{A}^i_{rec}$ is given by

$$\mathcal{A}^i_{diff} = \left(\mathcal{A}_{mod} \bigcup \mathcal{A}^i_{rec}\right) - \left(\mathcal{A}_{mod} \bigcap \mathcal{A}^i_{rec}\right). \tag{8.22}$$

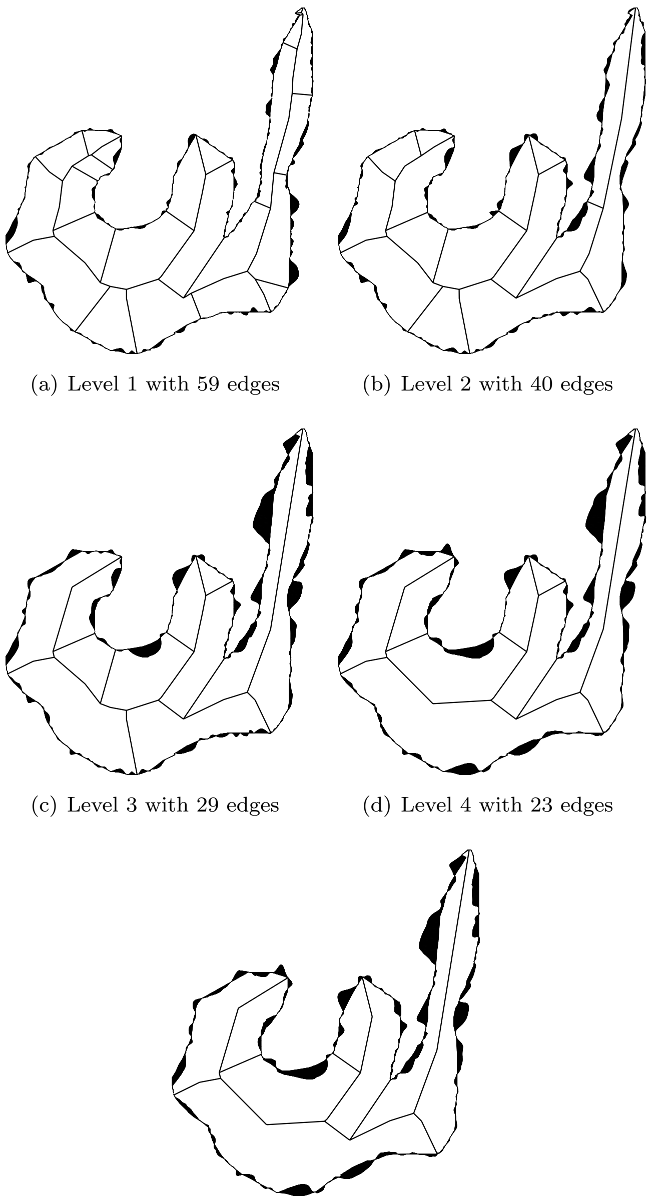
Figure 8.12 shows $\mathcal{A}^i_{diff}$ for different levels of hierarchy. Another important indicator of reconstruction accuracy is the ratio of areas $\mathcal{A}^i_{rec}/\mathcal{A}_{mod}$. Both of these measures are shown in Table 8.1. The tests show that with better sampling, the reconstruction accuracy increases. Also the ratio of areas indicate that the reconstructed curve has a slightly larger area than the original curve.

Table 8.1: *Reconstruction accuracy with respect to areas.*

		$\mathcal{A}_{mod} = 0.3228$			
Level	Edges	$\mathcal{A}^i_{rec}$	$\mathcal{A}^i_{diff}$	$\%\left(\frac{\mathcal{A}^i_{diff}}{\mathcal{A}_{mod}}\right)$	$\frac{\mathcal{A}^i_{rec}}{\mathcal{A}_{mod}}$
1	59	0.3266	0.0123	3.8	1.0119
2	40	0.3247	0.0168	5.2	1.0058
3	29	0.3279	0.0297	9.2	1.0160
4	23	0.3287	0.0333	10.3	1.0182
5	21	0.3283	0.0349	10.8	1.0171

The *Hausdorff distance* is a good measure of the distance between two manifolds [96]. Hausdorff distance, d_H, between two curves L and L' is given by

$$d_H(L, L') = \sup_{x_0 \in L} \inf_{x_1 \in L'} d(x, x'), \tag{8.23}$$


(a) Level 1 with 59 edges (b) Level 2 with 40 edges

(c) Level 3 with 29 edges (d) Level 4 with 23 edges

(e) Level 5 with 21 edges

Figure 8.12: *Skeleton hierarchy and reconstruction accuracy.*

where $d(\cdot,\cdot)$ is an appropriate metric for measuring distance between two points in a metric space. A mean value of the Hausdorff distance can be defined as [6]

$$d_m(L, L') = \frac{1}{|L|} \int_{x \in L} \inf_{x' \in L'} d(x, x') \mathrm{d}L, \tag{8.24}$$

where $|L|$ is the length of the curve L. A relative value of this distance is computed with respect to the bounding box of the object, as shown in Table 8.2. Also shown in the table is the length ratio of the reconstructed curve with respect to the original object contour. Here again, the tests show that a better sampling increases the accuracy of the reconstruction. However, we must point out that a topologically correct reconstruction is achievable with as few cutting lines as there are salient features in the object.

Table 8.2: *Reconstruction accuracy with respect to lengths.*

Level	Edges	$\%\left(\dfrac{d_H}{L_{diag}}\right)$	$\dfrac{L^i_{rec}}{L_{mod}}$
1	59	0.75	1.0842
2	40	0.82	1.0919
3	29	1.02	1.1227
4	23	1.08	1.1191
5	21	1.11	1.1186

8.5.2 Comparison

A comparison of our reconstruction with the results of Memari and Boissonnat [89] shows some of the shortcomings of their method that can be overcome with a reconstruction using continuous deformations. Reconstruction method proposed by Memari and Boissonnat is derived from the Delaunay complex of the cross sections. The reconstruction curve (see Figure 8.13) is only C^0 and misses some of the high curvature regions of the original object boundary.

We produce comparative statistics for our reconstruction with the method by Memari and Boissonnat [89]. The measures used for comparison are based on area and length of the reconstructed curve as introduced in the previous subsection. Table 8.3 shows reconstruction accuracy of three methods for the set of intersection lines shown in Figure 8.3. Here, Homotopy$_1$ refers to the reconstruction using continuous deformations with no tangent alignment and

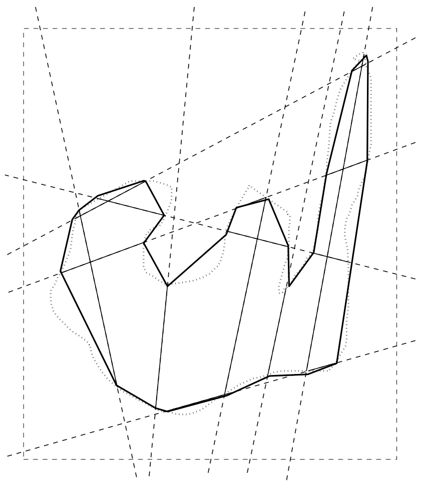


Figure 8.13: *Piecewise linear reconstruction using algorithm by [89].*

Homotopy$_2$ refers to the reconstruction resulting from tangent alignment, both with $\psi(x) = x$. Since [89] results in a piecewise linear reconstruction, the area (possibly) and length of the curve are underestimated. The ratio of absolute difference of area with the area of the model is low for Memari's reconstruction that matches up with Homotopy$_2$ reconstruction, but the relative Hausdorff measure goes bad and turns out to be more than double of that obtained with either of the Homotopy based reconstructions. Further, the relative ratio of the lengths of the reconstruction and the original object shows that homotopy based reconstructions perform better with estimating the length of the object.

Table 8.3: *Comparison of reconstructions.*

Method	$\%\left(\frac{\mathcal{A}_{diff}}{\mathcal{A}_{mod}}\right)$	$\frac{\mathcal{A}_{rec}}{\mathcal{A}_{mod}}$	$\%\left(\frac{d_H}{L_{diag}}\right)$	$\frac{L_{rec}}{L_{mod}}$
Memari	14.6025	0.9652	3.9323	0.9149
Homotopy$_1$	17.5470	1.0407	1.4831	1.0256
Homotopy$_2$	14.6690	1.0425	1.3243	1.0303

8.6 Conclusion

In this work, we have presented a novel method of curve reconstruction from arbitrary cross sections in a planar setting. The presented algorithm uses continuous deformations to reconstruct the object smoothly. We also introduced generalized barycentric coordinates for polygons defined on its edges. The presented method is general in nature and can be applied to higher dimensions.

Acknowledgement

The authors would like to thank Steen Markvorsen from Mathematics department at the Technical University of Denmark for fruitful discussions on the topic.

Chapter 9

Homotopic object reconstruction using natural neighbor barycentric coordinates

O. Sharma[†], and F. Anton[†]

[†]Department of Informatics and Mathematical Modelling,
Technical University of Denmark, Lyngby, 2800, Denmark

Abstract

One of the challenging problems in computer vision is object reconstruction from cross sections. In this paper, we address the problem of 2D object reconstruction from arbitrary linear cross sections. This problem has not been much discussed in the literature, but holds great importance since it lifts the requirement of order within the cross sections in a reconstruction problem, consequently making the reconstruction problem harder. Our approach to the reconstruction is via continuous deformations of line intersections in the plane. We define Voronoi diagram based barycentric coordinates on the edges of n-sided convex polygons as the area stolen by any point inside a polygon from the Voronoi regions of each open oriented line segment bounding the polygon. These allow us to formulate homotopies on edges of the polygons from which the underlying object can be reconstructed. We provide results of the reconstruction including the necessary derivation of the gradient at polygon edges.

9.1 Introduction

Object reconstruction from cross sections is a well known problem. Generally a spatial ordering within the cross sections aids reconstruction. We consider the problem of reconstructing an object from arbitrary linear cross sections. Such cross sectional data can be obtained from many physical devices. An example is acoustic probes that obtain range information of the object by sending an acoustic pulse.

The problem of reconstruction from arbitrary cross sections has been studied by [125, 83, 89]. Sidlesky et al. [125] define sampling conditions on the reconstruction, while in our reconstruction algorithm, we allow the sampling to be sparse. Methods proposed by Liu et al. [83], and Memari and Boissonnat [89] are both based on Voronoi diagrams. Memari and Boissonnat also provide rigorous proof of their reconstruction. Our approach to reconstruction considers the "presence" or "absence" of information along any intersecting line. This is in contrast to [125], where the authors consider that a line not intersecting the object does not contribute to the reconstruction. In our algorithm, such a line is considered to contribute to the reconstruction by defining a linear section, no part of which belongs to the reconstruction.

Memari and Boissonnat [89] provide a topological reconstruction method utilizing the Delaunay triagulation of the set of segments of intersecting lines. They claim an improvement over the method by Liu et al. [83] by producing reconstructions that are not topologically effected by lines that do not intersect the object under consideration. Their reconstruction boundary, however, is a piecewise linear approximation of the boundary of the original object. In this work, we produce smooth reconstruction of the object via continuous deformations. Therefore, we anticipate better reconstruction accuracy compared to the work by Memari and Boissonnat [89].

This paper is organized as follows. Section 9.2 defines the reconstruction problem mathematically. We introduce the concept of homotopy continuation in section 9.3 followed by the main reconstruction algorithm in section 9.4. We discuss our Voronoi diagram based edge barycentric coordinates on convex polygons here and provide details of our homotopy based reconstruction algorithm. Finally we provide results of the reconstruction.

9.2 Problem definition

Given a set of lines $\{\mathcal{L}_i : i \in [0, n-1]\}$ in a plane, intersecting an object $\mathcal{O}$ to generate segments $\{\mathcal{S}_{i,j} : j \in [0, m_i - 1]\}$, the problem of object reconstruction from arbitrary linear cross sections is to reconstruct object $\mathcal{R}$ from $\mathcal{S}_{i,j}$ such that

$$\mathcal{L}_i \bigcap \mathcal{O} = \mathcal{L}_i \bigcap \mathcal{R}, \tag{9.1}$$

and $\mathcal{R}$ is similar to $\mathcal{O}$. In this context we impose no restriction on the ordering or arrangement of the intersecting lines. One desired trait of the reconstruction is smoothness, and we will show that the proposed method of continuous deformations results in a reconstruction that is at least C^1.

In the next section, we define homotopy continuation that forms the basis of our reconstruction algorithm.

9.3 Homotopy continuation

Homotopy is concerned with identification of paths between objects that can be continuously deformed into each other. The history of study of homotopy dates back in the late 1920's when the the homotopy theory was formalized.

Definition 9.1 Let $f : X \mapsto Y$ and $g : X \mapsto Y$ be two continuous maps between topological spaces X and Y. These maps are called *homotopic*, $f \simeq g$, if there is a *homotopy* or a continuous map $\mathcal{H} : X \times [0,1] \mapsto Y$ between them, such that $\mathcal{H}(x, 0) = f(x)$ and $\mathcal{H}(x, 1) = g(x)$ for all $x \in X$.

Therefore, we can write the homotopy $\mathcal{H}_\lambda : X \mapsto Y$ as

$$\mathcal{H}_\lambda(x) = \mathcal{H}(x, \lambda), \tag{9.2}$$

and thus, $\mathcal{H}_0 = f$ and $\mathcal{H}_1 = g$. One can visualize how the deformation $\mathcal{H}_\lambda$ continuously takes f to g (see Figure 9.1) by varying the parameter λ.

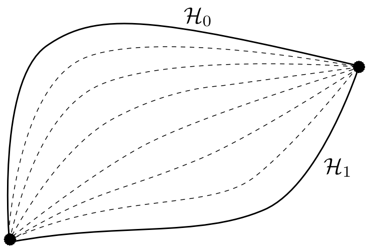


Figure 9.1: *Continuous deformation.*

One can impose additional constraints on the deformation path. For example, a specific constraint on fixed endpoints leads to homotopy of paths. For two pairs of homotopic maps $X \xrightarrow[f \simeq g]{} Y \xrightarrow[\bar{f} \simeq \bar{g}]{} Z$, the compositions $\bar{f} \circ f$ and $\bar{g} \circ g$ are also homotopic via the composition $\bar{\mathcal{H}}_\lambda \circ \mathcal{H}_\lambda$. Further, for two pairs of homotopic maps $f_i \simeq g_i : X_i \to Y_i, i = 1, 2$, the maps $f_1 \times f_2$ and $g_1 \times g_2$ from $X_1 \times X_2$ into $Y_1 \times Y_2$ are also homotopic via $\mathcal{H}_\lambda^{(1)} \times \mathcal{H}_\lambda^{(2)}$, in which case it is called a *product homotopy* [66].

Continuous deformations have been successfully used to solve non-linear system of equations that are otherwise hard to solve. A homotopy tries to solve a difficult problem with unknown solutions by starting with a simple problem with known solutions. Stable predictor-corrector and piecewise-linear methods for solving such problems exist (see Allgower and Georg [3]). The system $\mathcal{H}(x, \lambda) = 0$ implicitly defines a curve or one-manifold of solution points as λ varies in $[0, 1]$ and x is fixed.

Given smooth $\mathcal{H}$ and existence of $u_0 \in \mathbb{R}^{N+1}$ such that $\mathcal{H}(u_0) = 0$ and $\text{rank}(\mathcal{H}'(u_0)) = N$, there exists a smooth curve $c : \alpha \in J \mapsto c(\alpha) \in \mathbb{R}^{N+1}$ for some open interval J containing zero such that for all $\alpha \in J$ (Allgower and Georg [3])

1. $c(0) = u_0$,
2. $\mathcal{H}(c(\alpha)) = 0$,
3. $\text{rank}(\mathcal{H}'(c(\alpha))) = N$,
4. $c'(\alpha) \neq 0$.

In this work, we use homotopy or continuous deformations for object reconstruction. This is discussed in the next section.

9.4 Reconstruction algorithm

Starting with a set of cross sections $\{\mathcal{S}_{i,j}\}$ for lines $\{\mathcal{L}_i\}$ in a plane, we restrict reconstruction in the bounding box $\mathcal{B}_{box}$ of the cross sections. The set of lines $\{\mathcal{L}_i\}$ partition $\mathcal{B}_{box}$ into a set of convex polygons $\{\mathcal{G}_k : k \in [0, p-1]\}$. This is shown in Figure 9.2 where $\mathcal{O}$ is drawn dotted, the set of lines are shown dashed with the cross sections as thick solid lines, and the boundary of the bounding box shown dashed. Our reconstruction algorithm consists of assigning a homotopy $\mathcal{H}_k$ to every $\mathcal{G}_k$. The reconstruction is then obtained as

$$\mathcal{R} = \bigcup_k \{(x, y) : \mathcal{H}_k = 0\}. \tag{9.3}$$

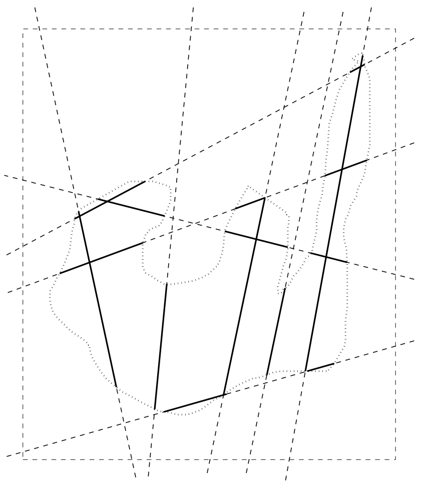


Figure 9.2: *A set of lines intersecting an object (dotted).*

A homotopy can be seen as a smooth transition from one map to another. We can extend this definition to multiple maps by defining a homotopy in multiple variables

$$\mathcal{H}(\lambda_0, \lambda_1, \cdots, \lambda_{s-1}) = \sum_{t=0}^{s-1} f_t \lambda_t, \tag{9.4}$$

with

$$\sum_{t=0}^{s-1} \lambda_t = 1. \tag{9.5}$$

9.4.1 Edge maps

Using (9.4), a smooth map can be defined over $\mathcal{G}_k$ for a choice of maps $\{f_t : t \in [0, s_k - 1]\}$ defined on s_k edges of $\mathcal{G}_k$. Let these maps be called *edge maps.* For continuity across all polygons, the definition of the edge maps must be consistent. Since, polygon edges are a subset of the cross section lines, it suffices to define edge maps over $\{\mathcal{L}_i\}$.

An edge map f_i should completely describe the boundary, interior and exterior of the intersection of $\mathcal{L}_i$ with $\mathcal{O}$. To define f_i, we associate a local coordinate system with each line $\mathcal{L}_i$ whose x-axis measures distance r along it from a chosen origin. Given abscissae $r_q, q \in [0, 2m_i - 1]$ of the intersections $\mathcal{S}_{i,j}$, we define the corresponding edge map as a piecewise quadratic polynomial

$$f_i(r) = \sum_{q=0}^{2m_i-2} \frac{\alpha_q(-r^2 + r(r_q + r_{q+1}) - r_q r_{q+1}))}{(r_{q+1} - r_q)}, \tag{9.6}$$

where α_q is the positive gradient $\left|\frac{\mathrm{d}\, f}{\mathrm{d}\, r}\right|_{r=r_q}$ defined as

$$\alpha_q = (-1)^{q+1}\alpha_0, \tag{9.7}$$

with α_0 being a chosen positive slope at r_0. Figure 9.3 illustrates such an edge map.

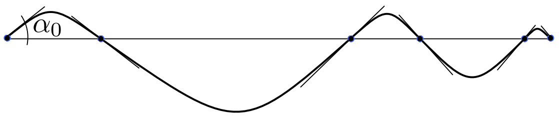


Figure 9.3: *Piecewise quadratic function as an edge map.*

9.4.2 Barycentric coordinates

It is natural to consider barycentric coordinates of a polygon as homotopy variables because of the two useful properties that they offer. Barycentric coordinates span a complete polygon and are a partition of unity. Traditional barycentric coordinates for triangles (and simplices in general) are defined by its vertices. Relevant generalizations of barycentric coordinates to n-sided polygons were provided by Wachspress [138] and later by Meyer et al. [90]. In the current context, we define barycentric coordinates in terms of the edges of a polygon rather than the vertices. Such a definition allows us to apply the concepts developed so far to associate a suitable homotopy to a polygon. We define Voronoi diagram based barycentric coordinates of edges for an n-sided polygon.

An interesting class of barycentric coordinates can be derived from Voronoi diagram of line segments and a point. Consider again a polygon $\mathcal{G}$ and a point p inside it. The Voronoi region of a point inside a polygon is a closed region of piecewise parabolic arcs as shown in Figure 9.4. From the Voronoi diagram of the edges $\{e_i\}$ in a polygon, introduction of a point p steals an area from two or more existing Voronoi regions. If the stolen area for any edge e_i and p is denoted by $\mathcal{A}_i$, the barycentric coordinates for the edge can be written as

$$\lambda_i = \frac{\mathcal{A}_i}{\sum\limits_{j=0}^{s-1} \mathcal{A}_j}, i \in [0, s-1], \tag{9.8}$$

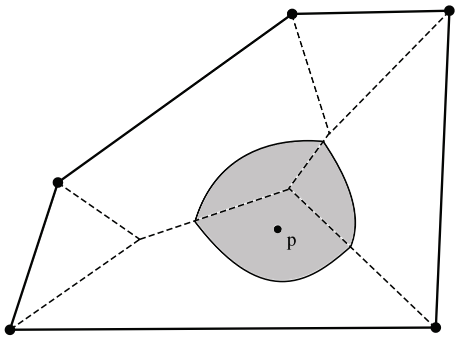


Figure 9.4: *Voronoi diagram of polygon and a point inside.*

Area $\mathcal{A}_i$ can be computed as the area between the parabolic arc and involved angle bisectors. Again, the barycentric coordinates defined in this way satisfy positivity, partition of unity and continuity. In the limiting case as p approaches one of the sides e_k, the stolen area $\mathcal{A}_k$ becomes very small, but simultaneously areas $\mathcal{A}_{i,i\neq k}$ become smaller (and eventually zero) by a rate higher than that of the former. Thus as p approaches e_k, $\lambda_k \to 1$, and $\lambda_{i,i\neq k} \to 0$.

9.4.3 Homotopy

Equipped with the above defined edge maps and barycentric coordinates for any polygon $\mathcal{G}_k$, we define a homotopy $\mathcal{H}_k$ in s_k variables as

$$\mathcal{H}_k(p) = \sum_{i=0}^{s_k-1} f_i\left(d_i(p)\right)\lambda_i(p), \tag{9.9}$$

where $d_i(p)$ is the distance along line $\mathcal{L}_k$ from a chosen origin O_k on it until the foot of the perpendicular from point p . We can write

$$d_i(p) = ||O_k - p_i|| + (p - p_i)^T \frac{(p_{i+1} - p_i)}{l_i}, \tag{9.10}$$

where p_{i+1} and p_i are two end points of an edge e_i of $\mathcal{G}_k$ lying on $\mathcal{L}_k$. Homotopy (9.9) continuously deforms edge maps f_i within the polygon and thus generates a smooth field. It can be seen as a linear combination of edge maps f_i with barycentric coordinates λ_i. We can further extend this so called *linear homotopy* to a *non-linear homotopy* as

$$\mathcal{H}_k(p,\eta) = \sum_{i=0}^{s_k-1} f_i\left(d_i(p)\right)\lambda_i(p)^{\eta}, \tag{9.11}$$

Across polygons, the homotopy (9.9) is continuous and at least C^1 smooth. We prove this for a planar triangulation in Appendix B. Results for a polygonal tessellation directly follow from the proof.

At all the intersection points of the lines with the boundary of the object, $q : \{\mathcal{L}_i \bigcap \partial\mathcal{O}\}$, the generated curve $\mathcal{H}^{-1}(0)$ is orthogonal to lines $\mathcal{L}_i$. Therefore, the resulting reconstruction is somewhat unnatural. Given normals at the intersection points q (which is the case with many range scanning physical devices), we propose a tangent alignment scheme for the resulting curve by locally warping the domain of the homotopy field.

9.4.4 Tangent alignment using local space rotations

Given unit normals $\widehat{\mathcal{N}}$ at intersection points q, the reconstruction can be constrained to be normally aligned to these normals at these points. We enforce this constraint by local space rotations around points q. The reconstruction $c = H^{-1}(0)$ is orthogonal to the intersecting lines at q. Starting with a point p in the neighborhood of one of the points p_c of q lying on $\mathcal{L}_k$, we rotate p

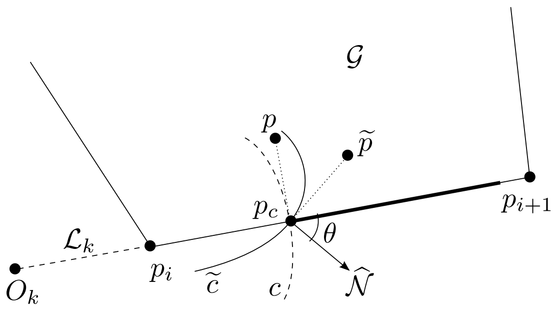


Figure 9.5: *Rotation of a point for tangent alignment.*

about p_c by an angle $-\theta$ to give point $\widetilde{p}$ in the plane (see Figure 9.5). The angle θ is chosen to be the signed angle between $\widehat{\mathcal{N}}$ and $\nabla\mathcal{L}_k$ at the point p_c. The resulting homotopy $\widetilde{\mathcal{H}}$ for a polygon $\mathcal{G}$ can be written as

$$\widetilde{\mathcal{H}} = \sum_{i=0}^{s_k-1} f_i\left(\widetilde{d_i}\right)\widetilde{\lambda}_i, \tag{9.12}$$

where,

$$\widetilde{d_i} = d_i(\widetilde{p}) = ||O_k - p_i|| + (\widetilde{p} - p_i)^T \frac{(p_{i+1} - p_i)}{l_i}, \text{ and}$$

$$\widetilde{\lambda}_i = \lambda_i(\widetilde{p}) = \frac{\widetilde{\mathcal{A}}_i}{\sum_{j=0}^{s-1}\widetilde{\mathcal{A}}_j}, i \in [0, s-1].$$

The rotated point $\widetilde{p}$ can be written as

$$\widetilde{p} = p_c + \mathbf{R}(-\theta)(p - p_c), \tag{9.13}$$

where $\mathbf{R}$ is the rotation matrix

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{9.14}$$

The modified gradient of the homotopy field in the neighborhood of p_c can be now computed using chain rule

$$\nabla\widetilde{\mathcal{H}} = \sum_{i=0}^{s-1}\left(f_i'(\widetilde{d_i})\nabla\widetilde{d_i}\widetilde{\lambda}_i + f_i(\widetilde{d_i})\nabla\widetilde{\lambda}_i\right). \tag{9.15}$$

In the limit as point $p \to p_c$ (or the orthogonal distance to $\mathcal{L}_k$, $\mu_k \to 0$), using a similar derivation as in Appendix B, we can write

$$\lim_{\mu_k \to 0} \nabla \widetilde{\mathcal{H}} = f'_k(\widetilde{d}_i) \nabla \widetilde{d}_k \tag{9.16}$$

Computing the gradient of $\widetilde{d}_k$,

$$\begin{aligned}
\nabla \widetilde{d}_k &= \nabla \left((\widetilde{p} - p_k)^T \frac{(p_{k+1} - p_k)}{l_k} \right) \\
&= \nabla \left((p_c + \mathbf{R}(-\theta)(p - p_c) - p_k)^T \frac{(p_{k+1} - p_k)}{l_k} \right) \\
&= \mathbf{R}(-\theta)^T \frac{(p_{k+1} - p_k)}{l_k} \\
&= \mathbf{R}(\theta) \nabla d_k
\end{aligned} \tag{9.17}$$

Therefore,

$$\lim_{\mu_k \to 0} \nabla \widetilde{\mathcal{H}} = f'_k(\widetilde{d}_i) \mathbf{R}(\theta) \nabla d_k \tag{9.18}$$

We know that the gradient

$$\lim_{\mu_k \to 0} \nabla \mathcal{H} = f'_k(d_i) \nabla d_k. \tag{9.19}$$

From (9.19) and (9.18) it can be seen that in the limit $\mu_k \to 0$

$$\frac{\nabla \widetilde{\mathcal{H}}}{||\nabla \widetilde{\mathcal{H}}||} = \mathbf{R}(\theta) \left(\frac{\nabla \mathcal{H}}{||\nabla \mathcal{H}||} \right). \tag{9.20}$$

Therefore, we can achieve the desired rotation of the reconstruction curve by rotating the local coordinates around the points of interest in the opposite direction.

9.4.5 Smooth rotations of the reconstruction curve

In order to generate a smooth distortion $\widetilde{H}^{-1}(0)$ of the curve $H^{-1}(0)$ the neighborhoods of points q must be carefully chosen. A natural neighborhood for points in q is their respective Voronoi polygons. However, a constant rotation for all the points in a particular Voronoi region results in a discontinuous curve at the boundary of these polygons. Therefore, we seek a continuous weight function w_{p_i} inside a Voronoi region $\mathcal{V}_q(p_i)$ of any point p_i of q such that

$$\begin{aligned}
&w_{p_i}(p_i) = 1, \text{ and} \\
&w_{p_i}(\partial \mathcal{V}_q(p_i)) = 0.
\end{aligned} \tag{9.21}$$

These requirements on the weight function impose a smooth transition of rotations from one influence zone to another and ensure little or no rotation at points far away from the centers of rotations. For any point p inside $\mathcal{V}_q(p_i)$, consider its nearest neighbor $p_i^{(1)}$ and the next nearest neighbor $p_i^{(2)}$ from the set of generators q. Denote by $d_1(p)$ the distance between p and $p_i^{(1)}$ and by $d_2(p)$ the one between p and $p_i^{(2)}$. We can formulate the required weight function as

$$w_{p_i}(p) = \frac{d_2(p) - d_1(p)}{d_2(p) + d_1(p)}. \tag{9.22}$$

The first nearest neighbor $p_i^{(1)}$ is the generator point p_i of $\mathcal{V}_q(p_i)$. The second nearest neighbor $p_i^{(2)}$ can be found by computing the second order voronoi diagram of q. The weight function resulting from (9.22) is shown in Figure 9.6. We outline the complete algorithm in Algorithm 9.1.

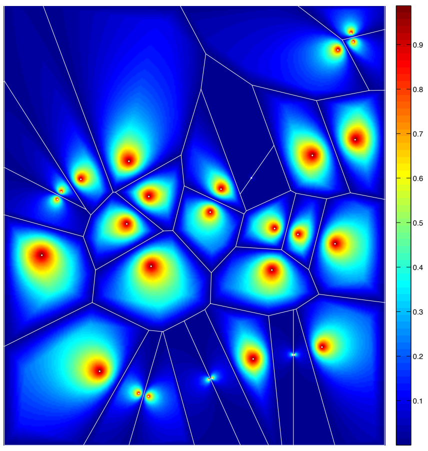


Figure 9.6: *Weights based on higher order Voronoi diagram.*

9.5 Results

We test our reconstruction method on a hand drawn curve intersected by a set of arbitrarily oriented lines. The lines yield a polygonal tessellation in the plane. We reconstruct the original curve from the intersections over the polygons of this tessellation.

```
**Input**: $\{\mathcal{S}_{i,j}\}$ on $\{\mathcal{L}_i\}$, $\widehat{\mathcal{N}}$ at $q$
**Output**: $\mathcal{R}$
Define an edge map $f_i$ on each $\mathcal{L}_i$ using $\{\mathcal{S}_{i,j}\}$
Partition $\mathcal{B}_{box}$ into polygonal tiles $\{\mathcal{G}_k\}$ with $\{\mathcal{L}_i\}$
Compute first order Voronoi diagram $\mathcal{V}_q^1$ of $q$
Compute second order Voronoi diagram $\mathcal{V}_q^2$ of $q$
Compute $w$ from $\mathcal{V}_q^1$ and $\mathcal{V}_q^2$
for $k \in [0, p-1]$ do
    Compute Voronoi diagram $\mathcal{V}_k$ of $\mathcal{G}_k$
    for $\mathbf{x} \in \mathcal{G}_k$ do
        Let $\mathbf{x}_c \in q$ be the generator of Voronoi polygon of $\mathbf{x}$
        $\widetilde{\mathbf{x}} \leftarrow \mathbf{x}_c + \mathbf{R}(-\theta)(\mathbf{x} - \mathbf{x}_c)$
        Compute $\{\lambda_i(\widetilde{\mathbf{x}})\}$ for all edges of $\mathcal{G}_k$ using $\mathcal{V}_k$
        Compute $d_i(\widetilde{\mathbf{x}})$ for all edges of $\mathcal{G}_k$
        Compute $\widetilde{\mathcal{H}}_k$
    end
end
$\widetilde{\mathcal{H}} \leftarrow \bigcup \widetilde{\mathcal{H}}_k$
$\mathcal{R} \leftarrow \ker \widetilde{\mathcal{H}}$
```

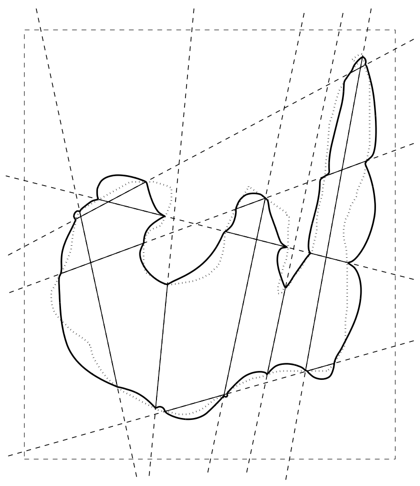
Algorithm 9.1: The reconstruction algorithm.

This reconstructed curve is guaranteed to pass through the end-points of the intersections. Figures 9.7(a), and 9.8(a) show results of our reconstruction for the linear and the non-linear homotopies. The reconstructed curve is always orthogonal to the intersecting lines. The orthogonality is not apparent in Figure 9.7(a) at a large scale, but is more visible for the non-linear homotopy with $\eta = 2$ in Figure 9.8(a).

We also show results of the reconstructed curve after applying local rotations at points q. The local rotations distort the original curve in the desired direction as seen in Figures 9.7(b), and 9.8(b).

9.6 Conclusion

In this work, we have presented a novel method of curve reconstruction from arbitrary cross sections in a planar setting. The presented algorithm uses continuous deformations to reconstruct the object smoothly. We also introduced generalized barycentric coordinates for polygons defined on its edges using the line and point Voronoi diagram. The presented method is general in nature and


(a) $\mathcal{H}^{-1}(0)$

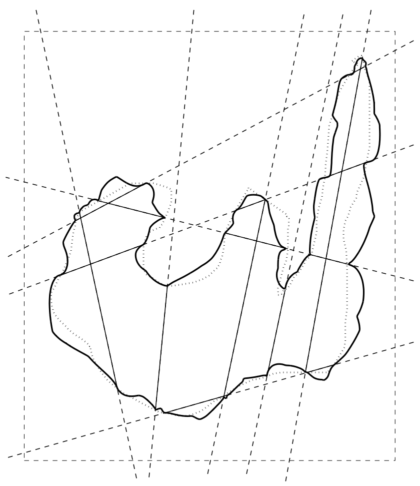
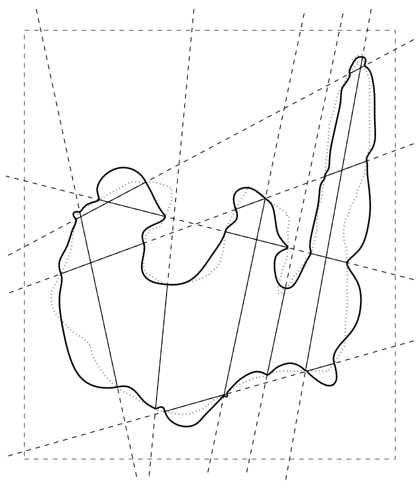

(b) $\widetilde{\mathcal{H}}^{-1}(0)$

Figure 9.7: *Reconstruction with linear homotopy.*


(a) $\mathcal{H}^{-1}(0)$

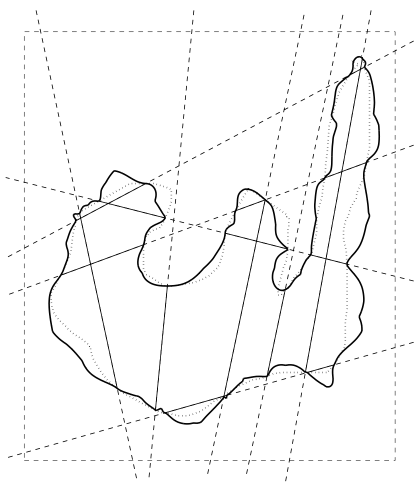

(b) $\widetilde{\mathcal{H}}^{-1}(0)$

Figure 9.8: *Reconstruction with non-linear homotopy.*

can be applied to higher dimensions.

Chapter 10

Discussion

This chapter puts the presented work in perspective of the objectives of this research and summarizes the thesis. A discussion is followed by a conclusion and future directions.

10.1 Discussion

The emphasis in this research has been to develop methods for object reconstruction via continuous deformations. While homotopy based methods have a solid topological background, they also provide natural insight into smoothly generating what is missing between the bits and pieces of available information. Various homotopies for smooth reconstruction are developed here and their suitability for reconstruction from underwater acoustic signals is shown. Validation of the reconstruction is shown on synthetic examples. The next subsections discuss some of the important points of this research work.

10.1.1 Comparison of level set method and homotopy reconstruction

Level set method has been suggested as effective means of segmentation of acoustic data in this research. Chapter 5 [115] introduces level set method and suggests a noise suppression scheme for reconstruction of moving fish in acoustic images. Level set methods are robust and produce topologically viable results. These are also the key arguments of using this method for segmentation in the suggested homotopy reconstruction framework. The level set method can also be seen as a reconstruction method and therefore, it is of interest to compare it with homotopy based reconstruction. Both of these methods have similarities and differences as outlined below.

Topological correctness

Both level set method and the suggested homotopy based reconstruction methods work in an implicit setting. Level set method evolves a deformable interface that aligns itself to various features of the image (in $\mathbb{R}^2$ and $\mathbb{R}^3$). Such an interface can be thought of as an elastic membrane that can be stretched to fit on to the desired object. Since, the interface is defined as a zero level set of an implicit function, topological changes are very well handled during deformation. On the other hand, Homotopy based reconstruction methods presented here define the reconstruction as the zero level set of a homotopy field that is constructed based on successive input signals. Between any two signals, the defined homotopy smoothly reconstructs the information. While the two methods are fundamentally different in the way a solution is obtained, they both benefit from the implicit setting they operate in.

Geometric smoothness

Traditional level set method obtain a solution of a time varying PDE in a tri-linear fashion by discretizing the equation and updating the implicit field. Therefore the resulting surface is C^0. Bajaj et al. [10, 11, 14] suggest a C^2 smooth solution to the level set equation that updates the implicit field by using a tri-cubic spline representation for the higher order terms in the level set PDE. For the homotopy based methods, C^0, C^1, and C^2 smooth homotopies are suggested in Chapter 7 [117]. The smoothness of the homotopy can be increased to any desired order by increasing the degree of the polynomial embedded in the beam (as a beam function) and by increasing the order of the homotopy via the homotopy parameter λ. Therefore, a homotopy based reconstruction can be made as smooth as desired.

Choice of parameters

Extraction of useful features in the level set framework rests on a suitable choice of parameters that keep a balance between the smoothness of the resulting surface and extraction of sharp features of objects. In the homotopy reconstruction framework, the smoothness is built into the reconstruction via choice of beam functions and the homotopy parameter. Sharp features result from singularities in the solution to the homotopy system. The present choice of piecewise quadratic beam functions is discussed in next subsection.

Computational burden

Level set methods are computationally expensive since an iterative solution to the governing PDE is required until convergence is reached. In the traditional level set method, the underlying implicit function is also required to be close to a signed distance field. Thus, a single solution step in the iteration involves a PDE update step followed by a redistancing (or reinitialization) step to keep the implicit a signed distance field. Redistancing is generally performed by solving (iteratively) the Eikonal equation [101] or by employing the Fast Marching method [134], for which efficient narrow-band implementations exist. Even though the PDE evolution is inherently parallel in nature, volumetric level set methods can still be slow due to their iterative nature. This is in contrast to the homotopy reconstruction method that involves formulating homotopies for pairs of beams and deriving a homotopy field from these. The method is non-iterative in nature and once the homotopy field is computed, the surface is extracted in a single step. A complexity analysis of the homotopy based reconstruction algorithm is discussed in section 7.7. Thus, the computational cost of the homotopy based reconstruction is lower and can be made real-time with the use of GPU assisted reconstruction.

10.1.2 Choice of beam functions in homotopy reconstruction

Piecewise-quadratic functions were considered in Chapter 7 as the choice of beam functions that constitute initial and terminal homotopy maps. This choice reflects the facts that quadratic equations are simple to solve analytically, a piecewise quadratic function is computationally inexpensive to formulate (i.e., the coefficients of piecewise quadratic functions are trivial to derive), and such a beam function has only one extrema between any two roots. These properties make sure that the reconstruction curve can be decomposed into parts containing either

- monotonic chains of the reconstructed curve (or a surface in $\mathbb{R}^3$) that start from the initial map and end at the terminal map, or
- simple curves that start and end at the initial map (or the terminal map) with exactly one extrema in between the curve branch.

This is shown in Figure 10.1 with critical points marked in red. Instead of a piecewise quadratic polynomial, a higher order piecewise-polynomial is also possible.

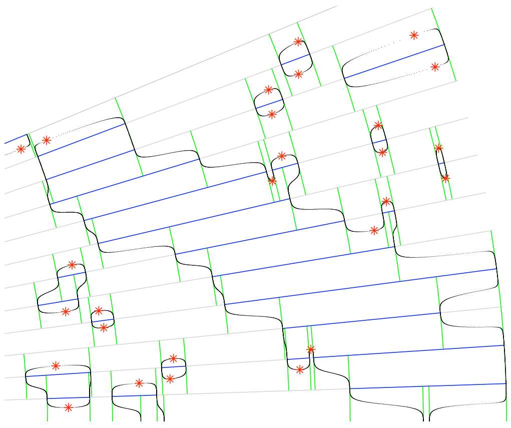


Figure 10.1: *Decomposition of the reconstruction curve. Points marked in red indicate the location of extrema.*

10.1.3 Parallel nature of algorithms

It is worth noting that the developed algorithms are highly parallel in nature and are suitable for computation on a GPU or in a parallel CPU environment. This is crucial for near real-time performance of the method to be usable in commercial systems (for example in sonar systems in ships). At the moment, Aquanaut provides an OpenMP parallelization of the reconstruction routines. CUDA based GPU solver has been developed for the level set segmentation of volumetric images.

10.1.4 Generalization to higher dimensions

Chapter 7 presents reconstructions in $\mathbb{R}^2$ and $\mathbb{R}^3$, while Chapters 8 and 9 show the results of reconstruction from arbitrary cross sections in $\mathbb{R}^2$. The presented reconstruction methods are generic in nature and can be easily generalized to higher dimensions.

A physically realistic generalization of the homotopy reconstruction algorithm [117] is in space-time reconstruction of objects. Such a problem is very commonly encountered in reconstructing moving objects. This problem can be topologically challenging if the objects under consideration also change their shapes (non-rigid objects). For a time varying signal $S(\mathbf{x}, t) : \mathbb{R}^3 \times \mathbb{R} \mapsto \mathbb{R}$, time-varying beam functions $f(\mathbf{x}, t) : \mathbb{R}^3 \times \mathbb{R} \mapsto \mathbb{R}$ can be assigned to the beams. A homotopy then takes the generalized form of (7.14)

$$\mathcal{H}_{i,j,k}(\mathbf{x}, t, \alpha, \beta, \gamma) = \mathcal{H}_{i,j}(\mathbf{x}, t, \alpha) \otimes \mathcal{H}_{j,k}(\mathbf{x}, t, \beta) \otimes \mathcal{H}_{k,i}(\mathbf{x}, t, \gamma).$$

An interesting generalization of the problem of reconstruction from arbitrary cross sections [118] is 3D reconstruction from arbitrary cutting planes. These cutting planes partition the domain of computation into polyhedra, and also embed the intersection with the object. Similar to the approach suggested in Chapters 8 and 9 [118, 116], a function $f : \mathbb{R}^2 \mapsto \mathbb{R}$ can be embedded in a cutting plane such that $\ker(f)$ represents the boundary of this intersection. An example of one such function is the signed distance function. Higher degree polynomial functions can be designed based on this distance function. A multivariate homotopy can then be constructed from the functions defined previously on the faces of a polyhedron. A possible parameterization of such a homotopy is with respect to the orthogonal distance of any point inside the polyhedron to the polyhedron faces (see Figure 10.2). Parameterization in terms of the Voronoi volume (a volume consisting of a paraboloid face and planar faces) stolen by a point inside the polyhedron is another choice. A zero level set of the derived homotopy field provides a reconstruction surface.

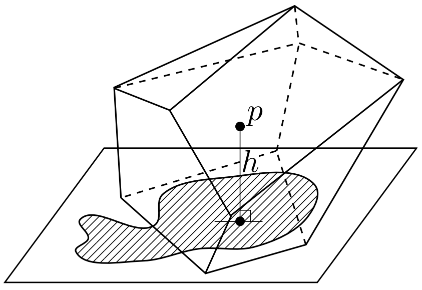


Figure 10.2: *Reconstruction from arbitrary cutting planes in* $\mathbb{R}^3$.

10.2 Conclusion

The main goal set forth in this research was to apply the method of continuous deformation to object reconstruction from acoustic signals. The project started with a survey of various techniques for volumetric segmentation. Level set method is explored as an effective means of segmentation and is adapted for acoustic volume segmentation as shown in Chapter 5 [115]. For large volume processing, CUDA based streaming GPU solver is designed. The solver is further enhanced with a higher-order, and multi-domain level set method in as described in Chapter 6 [123]. Results on medical tomography and microscopy volumes are shown and reconstruction accuracy is discussed.

The linear and non-linear homotopy methods for reconstruction were first designed and subsequently refined iteratively to derive the higher order homotopy methods in Chapter 7 [117]. The linear homotopy surface is piecewise monotonic, but is only C^0 smooth, while the non-linear surface is C^{n-1} smooth but suffers from the stair-casing effect in the reconstructed surface. The local nature of the linear homotopy is improved by extending it to the cubic spline homotopy which is global in nature. A computationally faster form of the cubic spline homotopy is formulated as the B-spline homotopy. Although the cubic spline homotopy generates a smooth C^2 surface with no artifacts at the border of two beams, it is not piecewise monotonic. This problem manifests itself as unnatural bumps on the surface near sharp corners (due to undesired extrema on the surface). This problem is addressed in the design of shape preserving or monotone cubic homotopy. The generated surface is C^1 smooth at the beam boundaries and C^2 smooth otherwise.

An offshoot of reconstruction from linear cross sections was the problem of reconstruction from arbitrary cross sections addressed in Chapters 8 and 9 [118, 116]. Such a problem is a natural extension of the existing one since ordering in cross sections might not always be present. This problem is discussed in context of smooth 2D object reconstruction. The interesting part here is to design barycentric coordinates based on edges of a polygon. Classical vertex based barycentric coordinates cannot be used here since the information of object intersection is associated with the polygon edges and not with the polygon vertices. It is proved that the reconstructed object is at least C^1 smooth. The boundary of the reconstruction is orthogonal to the intersecting lines. If information about the normals at the extremities of the intersections along beams is known, the orthogonality can be eliminated by means of local space rotations around these extremities. The suggested method is easily extensible to higher dimensions as discussed before in subsection 10.1.4.

10.3 Future directions

The developed reconstruction methods are applicable to real world sonar data and an immediate extension is to incorporate rotations and translations in the sonar instrument caused by vessel movement and turbulence in the water during a marine survey (see Figure 10.3). The motion of the vessel, assumed piecewise-

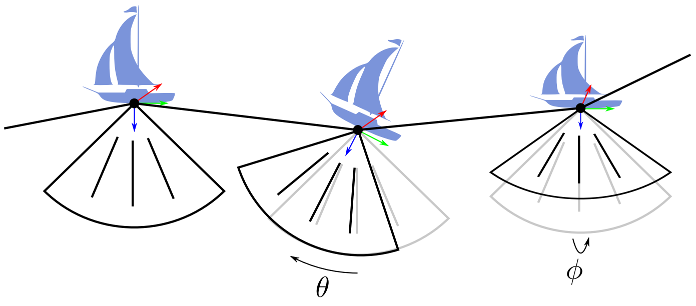


Figure 10.3: *Tilt and translation in real sonar systems.*

linear, for a set of sonar observations between any two successive frames can be parameterized in terms of

- the translation ($t \in [t_0, t_1]$) along the axis of motion,
- the three rotations ($\theta \in [\theta_0, \theta_1], \phi \in [\phi_0, \phi_1], \gamma \in [\gamma_0, \gamma_1]$) about the axis of motion, and orthogonal to it.

For a two parameter family of homotopy $\mathcal{H}(\alpha, \beta)$, $\beta \in [0, 1]$ parameterizes the homotopy within a sonar fan and α parameterizes the homotopy between successive sonar fans, where

$$\alpha = f(t, \theta, \phi, \gamma), \tag{10.1}$$

such that $f : \mathbb{R}^4 \mapsto [0, 1]$ is a bijection. Instead of a piece-wise linear motion profile, a spline representation can also be used.

In conclusion, this thesis provides a direction in the less explored tool of homotopy continuation for the much sought problem of object reconstruction. The prospect of this research is in emergence of better algorithms for reconstruction.

Appendix A

Smoothness in homotopies - Barycentric coordinates based on orthogonal distance

Proposition A.1 *For a polygon $\mathcal{G}$ in a planar tessellation , we show that the homotopy*

$$\mathcal{H}(p) = \sum_{i=0}^{s-1} f_i\left(d_i(p)\right)\lambda_i(p) = 0, \tag{A.1}$$

is at least C^1 for barycentric coordinates defined as

$$\lambda_i = \frac{1/\psi(h_i)}{\sum_{j=0}^{s-1} 1/\psi(h_j)}, i \in [0, s-1], \tag{A.2}$$

where $\psi : \mathbb{R} \to \mathbb{R}$ is a monotonically increasing smooth function with $\psi(0) = 0$, and h_i is the orthogonal distance from point p to any edge e_i of $\mathcal{G}$.

Proof. We can prove this by showing that (A.1) is C^0 and C^1.

1. $\mathcal{H} = 0$ is C^0.

From the properties of barycentric coordinates, we know that

$$\sum_{i=0}^{s-1} \lambda_i = 1.$$

This implies that at any edge e_k of $\mathcal{G}$, $\lambda_k = 1$ and $\lambda_{i,i \neq k} = 0$. Therefore, for any p lying on e_k, $\mathcal{H}(p) = f_k(d_k(p))$. The same holds for any other polygon in the tessellation and since the functions f_i are globally defined on lines L_i, $\mathcal{H} = 0$ is C^0.

2. $\mathcal{H} = 0$ is C^1.

In order to show that $\mathcal{H} = 0$ is C^1, we calculate the gradient of $\mathcal{H}$ at any e_k. For any point $p(x, y)$ inside $\mathcal{G}$ and an edge e_i connecting vertices $p_i(x_i, y_i)$ and $p_{i+1}(x_{i+1}, y_{i+1})$ of $\mathcal{G}$, we note that the distance d_i along line $\mathcal{L}_j$ (corresponding to e_i) until the foot of the perpendicular from p is given by

$$d_i = \frac{\triangle x_i (x - x_i) + \triangle y_i (y - y_i)}{l_i}, \tag{A.3}$$

where $\triangle x_i = (x_{i+1} - x_i)$, $\triangle y_i = (y_{i+1} - y_i)$, and l_i is the length of e_i. Further, the orthogonal distance h_i from p on to e_i is given by

$$h_i = \frac{\triangle y_i (x - x_i) - \triangle x_i (y - y_i)}{l_i}. \tag{A.4}$$

We can compute the derivatives of d_i and h_i as

$$\nabla d_i = \left(\frac{\partial d_i}{\partial x}, \frac{\partial d_i}{\partial y} \right) = \left(\frac{\triangle x_i}{l_i}, \frac{\triangle y_i}{l_i} \right) \tag{A.5}$$

$$\nabla h_i = \left(\frac{\partial h_i}{\partial x}, \frac{\partial h_i}{\partial y} \right) = \left(\frac{\triangle y_i}{l_i}, -\frac{\triangle x_i}{l_i} \right) \tag{A.6}$$

We compute the derivative of λ_i as defined in (A.2)

$$\begin{aligned}\frac{\partial \lambda_i}{\partial x} =& \frac{-1}{(\psi(h_i))^2} \frac{\psi'(h_i)}{\sum\limits_{j=0}^{s-1} 1/\psi(h_j)} \frac{\partial h_i}{\partial x} \\ &+ \frac{1}{\psi(h_i)} \frac{-\sum\limits_{j=0}^{s-1} \frac{-\psi'(h_j)}{(\psi(h_j))^2} \frac{\partial h_j}{\partial x}}{\left(\sum\limits_{j=0}^{s-1} 1/\psi(h_j)\right)^2} \\ =& -\lambda_i \frac{\psi'(h_i)}{\psi(h_i)} \frac{\partial h_i}{\partial x} + \lambda_i \sum_{j=0}^{s-1} \lambda_j \frac{\psi'(h_j)}{\psi(h_j)} \frac{\partial h_j}{\partial x} \qquad \text{(A.7)} \\ \frac{\partial \lambda_i}{\partial y} =& -\lambda_i \frac{\psi'(h_i)}{\psi(h_i)} \frac{\partial h_i}{\partial y} + \lambda_i \sum_{j=0}^{s-1} \lambda_j \frac{\psi'(h_j)}{\psi(h_j)} \frac{\partial h_j}{\partial y} \qquad \text{(A.8)}\end{aligned}$$

We can now compute the derivative of $\mathcal{H}$ w.r.t. x as

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial x} =& \sum_{i=0}^{s-1} \left(f_i'(d_i) \frac{\partial d_i}{\partial x} \lambda_i + f_i(d_i) \frac{\partial \lambda_i}{\partial x} \right) \qquad \text{(A.9)} \\ =& \sum_{i=0}^{s-1} \left(f_i'(d_i) \frac{\partial d_i}{\partial x} \lambda_i + f_i(d_i) \lambda_i \left\{ - \frac{\psi'(h_i)}{\psi(h_i)} \frac{\partial h_i}{\partial x} \right.\right. \\ &\left.\left. + \sum_{j=0}^{s-1} \lambda_j \frac{\psi'(h_j)}{\psi(h_j)} \frac{\partial h_j}{\partial x} \right\} \right).\end{aligned}$$

In order to check continuity of $\mathcal{H} = 0$ at e_k (or at the corresponding line $\mathcal{L}_{k'}$), we are interested in the limit $\lim_{h_k \to 0} \nabla \mathcal{H}$ from two sides of the $\mathcal{L}_{k'}$ shared by polygons $\mathcal{G}_1$ and $\mathcal{G}_2$. In $\mathcal{G}_1$, as $h_k \to 0^{(1)}$, we note that $T_k \to 0$, $\lambda_k \to 1$, $\lambda_{i, i \neq k} \to 0$. Thus,

$$\begin{aligned}\lim_{h_k \to 0^{(1)}} \frac{\partial \mathcal{H}}{\partial x} =& f_k'(d_k) \frac{\partial d_k}{\partial x} - f_k(d_k) \frac{\psi'(h_k)}{\psi(h_k)} \frac{\partial h_k}{\partial y} \\ &+ f_k(d_k) \frac{\psi'(h_k)}{\psi(h_k)} \frac{\partial h_k}{\partial y} \\ =& f_k'(d_k) \frac{\triangle x_k}{l_k}. \qquad \text{(A.10)}\end{aligned}$$

Therefore limit of the gradient $\nabla\mathcal{H}$ as $h_k \to 0$ is

$$\lim_{h_k\to 0} \nabla\mathcal{H} = \left(f_k'(d_k)\frac{\triangle x_k}{l_k}, f_k'(d_k)\frac{\triangle y_k}{l_k} \right). \tag{A.11}$$

For the line $\mathcal{L}_j$ passing through p_i and p_{i+1},

$$\mathcal{L}_j : \triangle y_i(x - x_i) - \triangle x_i(y - y_i) = 0, \tag{A.12}$$

the gradient is

$$\nabla\mathcal{L}_j = (\triangle y_i, -\triangle x_i). \tag{A.13}$$

In $\mathcal{G}_1$, we see that $\lim_{h_k\to 0} \nabla\mathcal{H} \perp \nabla\mathcal{L}_j$. On the other side of $\mathcal{L}_j$ in polygon $\mathcal{G}_2$, the limit of the gradient of $\mathcal{H}$ gives the same result. This shows that the gradients of $\mathcal{H}$ are equal from the two sides of $\mathcal{L}_j$ and are orthogonal to the gradient of $\mathcal{L}_j$.

Therefore, the reconstructed curve is at least C^1. □

Appendix B

Smoothness in homotopies - Barycentric coordinates based on stolen area

Proposition B.1 *For a triangle $\mathcal{T}$ in a planar triangulation , we show that the curve $\mathcal{H}^{-1}(0)$ defined by the homotopy (9.9),*

$$\mathcal{H}(p) = \sum_{i=0}^{2} f_i\left(d_i(p)\right)\lambda_i(p) = 0, \tag{B.1}$$

is at least C^1 for barycentric coordinates based on the stolen area $\mathcal{A}_i$ for any edge e_i of $\mathcal{T}$ and point p, defined as

$$\lambda_i = \frac{\mathcal{A}_i}{\sum_{j=0}^{2}\mathcal{A}_j}, i \in [0, 2]. \tag{B.2}$$

Proof. We can prove this by showing that $\mathcal{H}^{-1}(0)$ is C^0 and C^1.

1. $\mathcal{H}^{-1}(0)$ is C^0.

From the properties of barycentric coordinates, we know that

$$\sum_{i=0}^{2} \lambda_i = 1.$$

This implies that at any edge e_k of $\mathcal{T}$, $\lambda_k = 1$ and $\lambda_{i,i \neq k} = 0$. Therefore, for any p lying on e_k, $\mathcal{H}(p) = f_k(d_k(p))$. The same holds for any other triangle in the triangulation and since the functions f_i are globally defined on lines L_i, $\mathcal{H} = 0$ is C^0.

2. $\mathcal{H}^{-1}(0)$ is C^1.

In order to show that $\mathcal{H}^{-1}(0)$ is C^1, we calculate the gradient of $\mathcal{H}$ at any e_k. The derivative of $\mathcal{H}$ is

$$\nabla\mathcal{H} = \sum_{i=0}^{2} \left(f_i'(d_i)\nabla d_i \lambda_i + f_i(d_i)\nabla\lambda_i\right) \tag{B.3}$$

where d_i is the distance along line $\mathcal{L}_j$ corresponding to edge e_i of the triangle.

The gradient of λ_i from (B.2) can be calculated using the chain rule as

$$\begin{aligned}\nabla\lambda_i &= \frac{\nabla\mathcal{A}_i}{\sum_{j=0}^{2}\mathcal{A}_j} - \frac{\mathcal{A}_i}{\left(\sum_{j=0}^{2}\mathcal{A}_j\right)^2}\sum_{j=0}^{2}\nabla\mathcal{A}_j \\ &= \frac{\nabla\mathcal{A}_i}{\sum_{j=0}^{2}\mathcal{A}_j} - \lambda_i \sum_{j=0}^{2}\frac{\nabla\mathcal{A}_j}{\sum_{j=0}^{2}\mathcal{A}_j} \end{aligned} \tag{B.4}$$

Using (B.3) and (B.4),

$$\nabla\mathcal{H} = \sum_{i=0}^{2}\left(f_i'(d_i)\nabla d_i\lambda_i + f_i(d_i)\left[\frac{\nabla\mathcal{A}_i}{\sum_{j=0}^{2}\mathcal{A}_j} - \lambda_i\sum_{j=0}^{2}\frac{\nabla\mathcal{A}_j}{\sum_{j=0}^{2}\mathcal{A}_j}\right]\right) \tag{B.5}$$

To compute the gradient of $\mathcal{H}$ at the intersection of line $\mathcal{L}_j$ and the object, we must take the derivative at a point p in the limit as p approaches line $\mathcal{L}_j$. In this limit,

$$\begin{aligned} \lambda_k &\to 1 \\ \lambda_{i,i\neq k} &\to 0 \end{aligned} \tag{B.6}$$

Before the gradient in the limit can be evaluated, the behavior of $\nabla \mathcal{A}_i$ and $\mathcal{A}_i$ should be analyzed. To do so, we define several quantities as following.

Consider the triangle $\mathcal{T}$ shown in Figure B.1. Let the vertices of $\mathcal{T}$ be denoted by $p_0(x_0, y_0)$, $p_1(x_1, y_1)$, and $p_2(x_2, y_2)$, and the edges by $e_0 = (p_1 - p_0)$, $e_1 = (p_2 - p_1)$, and $e_2 = (p_0 - p_2)$. Let an edge e_i of $\mathcal{T}$ be parameterized by distance α along it

$$e_i : p = p_i + \alpha \frac{(p_{i+1} - p_i)}{l_i}, \tag{B.7}$$

where index i is to be taken in a circular sense in the triangle. The Voronoi diagram of edges of $\mathcal{T}$ divide it internally in three regions. We consider a point p lying in the Voronoi region of an edge e_i of $\mathcal{T}$ (see Figure B.1).

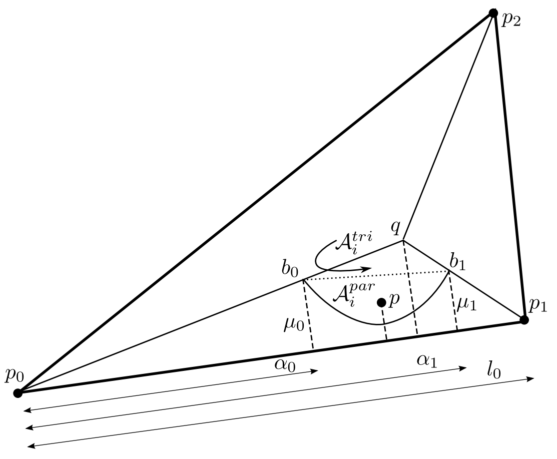


Figure B.1: *Parameterization along triangle edge.*

A parabola with focus at point p and directrix e_i is given by

$$\mathcal{P}_i : \mu(\alpha) = \frac{(\alpha - \alpha_p)^2}{2\mu_p} + \frac{\mu_p}{2}, \tag{B.8}$$

where (α, μ) form an orthonormal basis, and

$$\alpha_p = (p - p_i)^T \frac{(p_{i+1} - p_i)}{l_i}, \tag{B.9}$$

$$\mu_p = (p - p_i)^T \mathbf{M} \frac{(p_{i+1} - p_i)^T}{l_i}, \tag{B.10}$$

with

$$\mathbf{M} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \tag{B.11}$$

and l_i is the length of e_i.

Let the parabola $\mathcal{P}_i$ intersects the boundary of the Voronoi region at points $b_i(\alpha_0, \mu_0)$ and $b_{i+1}(\alpha_1, \mu_1)$ respectively. Denote by $q(\alpha_q, \mu_q)$ the incenter of $\mathcal{T}$. The parabola can intersect the two angle bisectors of the triangle $\mathcal{B}_i : \mu = k_0\alpha$, and $\mathcal{B}_{i+1} : \mu = k_1(l_i - \alpha)$, $k_i = \tan(\theta_i/2)$, in three possible ways

I. $\alpha_1 \leq \alpha_q$: both branches of the parabola intersect $\mathcal{B}_i$, and $\mu_m = k_0\alpha_m, m = \{0, 1\}$,

II. $\alpha_0 \geq \alpha_q$: both branches of the parabola intersect $\mathcal{B}_{i+1}$, and $\mu_m = k_1(l_i - \alpha_m), m = \{0, 1\}$, and

III. $\alpha_0 < \alpha_q$ and $\alpha_1 > \alpha_q$: the branches of the parabola intersect $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ respectively, and $\mu_0 = k_0\alpha_0$ and $\mu_1 = k_1(l_i - \alpha_1)$.

It is sufficient to treat one of these cases here. Considering case I, α_0 and α_1 are the roots of

$$\alpha^2 - 2\alpha(\alpha_p + k_0\mu_p) + (\alpha_p^2 + \mu_p^2) = 0. \tag{B.12}$$

The areas to compute the barycentric coordinates can be written as

$$\mathcal{A}_i = \mathcal{A}_i^{par} + \mathcal{A}_i^{tri} \tag{B.13}$$

where, $\mathcal{A}_i^{par}$ is the area enclosed between the parabolic arc and line connecting b_i and b_{i+1}, and $\mathcal{A}_i^{tri}$ is the area of the triangle connecting points b_i, b_{i+1}, and q. Using (B.8)

$$\begin{aligned} \mathcal{A}_i^{par} = & \frac{\mu_0 + \mu_1}{2}(\alpha_1 - \alpha_0) \\ & - \int_{\alpha_0}^{\alpha_1} \left(\frac{(\alpha - \alpha_p)^2}{2\mu_p} + \frac{\mu_p}{2} \right) \mathrm{d}\alpha, \end{aligned} \tag{B.14}$$

and

$$\begin{aligned}\mathcal{A}_i^{tri} = \frac{\mu_0+\mu_q}{2}(\alpha_q-\alpha_0) + \frac{\mu_1+\mu_q}{2}(\alpha_1-\alpha_q) \\ - \frac{\mu_0+\mu_1}{2}(\alpha_1-\alpha_0). \end{aligned} \tag{B.15}$$

Therefore,

$$\begin{aligned}\mathcal{A}_i = \frac{\mu_0+\mu_q}{2}(\alpha_q-\alpha_0) + \frac{\mu_1+\mu_q}{2}(\alpha_1-\alpha_q) \\ - \int_{\alpha_0}^{\alpha_1}\left(\frac{(\alpha-\alpha_p)^2}{2\mu_p}+\frac{\mu_p}{2}\right)\mathrm{d}\alpha \end{aligned} \tag{B.16}$$

Note that (B.16) holds for all the three cases above. Simplifying (B.16), we get

$$\begin{aligned}2\mathcal{A}_i &= (\mu_0+\mu_q)(\alpha_q-\alpha_0) + (\mu_1+\mu_q)(\alpha_1-\alpha_q) \\ &\quad - \left[\frac{(\alpha-\alpha_p)^3}{3\mu_p} - \mu_p\alpha\right]_{\alpha_0}^{\alpha_1} \\ &= (\mu_0+\mu_q)(\alpha_q-\alpha_0) + (\mu_1+\mu_q)(\alpha_1-\alpha_q) \\ &\quad - \frac{(\alpha_1-\alpha_p)^3-(\alpha_0-\alpha_p)^3}{3\mu_p} - \mu_p(\alpha_1-\alpha_0) \end{aligned} \tag{B.17}$$

Distance d_i in (9.9) can be written as

$$d_i(p) = ||O_j - p_i|| + \alpha_p, \tag{B.18}$$

where we know that p_i lies on $\mathcal{L}_j$ and O_j is the chosen origin on $\mathcal{L}_j$. We note the following derivatives

$$\begin{aligned}
2\nabla\mathcal{A}_i = {} & \nabla\alpha_0(-\mu_0-\mu_p)+\nabla\alpha_1(\mu_1+\mu_p) \\
& + \nabla\mu_0(-\alpha_0+\alpha_q)+\nabla\mu_1(\alpha_1-\alpha_q) \\
& + \frac{(\alpha_1-\alpha_p)^3-(\alpha_0-\alpha_p)^3}{3\mu_p^2}\nabla\mu_p \\
& - \frac{(\alpha_1-\alpha_p)^2(\nabla\alpha_1-\nabla\alpha_p)}{\mu_p} \\
& + \frac{(\alpha_0-\alpha_p)^2(\nabla\alpha_0-\nabla\alpha_p)}{\mu_p} \\
& - (\alpha_1-\alpha_0)\nabla\mu_p-\mu_p(\nabla\alpha_1-\nabla\alpha_0) \qquad \text{(B.19)}
\end{aligned}$$

$$\nabla d_i = \nabla\alpha_p = \frac{(p_{i+1}-p_i)}{l_i} \qquad \text{(B.20)}$$

$$\nabla\mu_p = \mathbf{M}\frac{(p_{i+1}-p_i)}{l_i} \qquad \text{(B.21)}$$

$$\nabla\alpha_m = \frac{\nabla\alpha_p(\alpha_m-\alpha_p)+\nabla\mu_p(\mu_m-\mu_p)}{(\alpha_m-\alpha_p-k_0\mu_p)}, m=\{0,1\} \qquad \text{(B.22)}$$

$$\nabla\mu_m = k_0\nabla\alpha_m,\ m=\{0,1\} \qquad \text{(B.23)}$$

In the limit $\mu_p \to 0$ for some edge e_k of $\mathcal{T}$,

$$\begin{aligned}
&\alpha_m \to \alpha_p,\ m\in\{0,1\} \\
&\mu_m \to k_0\alpha_p\ m\in\{0,1\}
\end{aligned} \qquad \text{(B.24)}$$

Consequently, $\mathcal{A}_i \to 0$, but $\mathcal{A}_{i,i\neq k}$ becomes 0 much faster than $\mathcal{A}_k$. Therefore, using (B.6) and (B.24), the gradient (B.5) in the limit is

$$\begin{aligned}
\lim_{\mu_p\to 0}\nabla\mathcal{H} &= f_k'(d_k)\nabla d_k + f_k(d_k)\left[\frac{\nabla\mathcal{A}_k}{\sum_{j=0}^{2}\mathcal{A}_j} - \frac{\nabla\mathcal{A}_k}{\sum_{j=0}^{2}\mathcal{A}_j}\right] \\
&= f_k'(d_k)\nabla d_k. \qquad \text{(B.25)}
\end{aligned}$$

Gradient of line e_k is

$$\nabla e_k = \mathbf{M}\frac{(p_{i+1}-p_i)}{l_i}. \qquad \text{(B.26)}$$

We note that

$$\begin{aligned}\left\langle \lim_{\mu_p \to 0} \nabla \mathcal{H}, \nabla e_k \right\rangle &= f_k'(d_k) \frac{(p_{i+1} - p_i)^T}{l_i} \mathbf{M} \frac{(p_{i+1} - p_i)}{l_i} \\ &= 0.\end{aligned}$$

A similar result can be shown for the gradient of the homotopy on the other side of $\mathcal{L}_j$. This implies that the reconstructed curve is orthogonal to the intersecting lines from either side. Therefore, the curve reconstruction is at least C^1.

APPENDIX C

Aquanaut - A homotopy reconstruction library

The homotopy continuation based algorithms presented in this work for reconstruction from sonar signals have been implemented as a set of libraries called *Aquanaut*. It implements the following homotopies for reconstruction

- Linear,
- Non-linear,
- Cubic spline, and
- Shape preserving.

The library comprises of the following components

- **Dynamic libraries**
  - *Homotopy* (libHomotopy): provides homotopy reconstruction algorithms.

  - *Raytracer* (libRaytracer): provides raytracing capabilities of basic geometric primitives for a simulation of reconstruction. Available primitives are: Sphere, Ellipsoid, Cuboid, Cone, Cylinder, and Torus. Dummy sonar classes provide containers for these geometric primitives.
  - *VTK I/O* (libVtkIO): provides basic read and write functionality for VTK volumetric images.

- **Aquanaut**
  A graphical user interface for the libraries and an integrated application for surface reconstruction for both real data and simulated scenes.
- **Plugins**: A collection of plugins for the application Aquanaut. At the moment, only following plugins are available
  - *Data import* (libDataImport): provides wizards and interfaces for import of data from Simrad MS70, RESON s7k, and Aquanaut smass (binary segmentation or characteristic function of signals) datasets.
- **External dependencies**
  - *GEL* (GEometric and Linear algebra tools) [7]: It is a framework for computer graphics and computer vision written mostly by Andreas Bærentzen and colleagues at DTU Informatics, The Technical University of Denmark.

A class diagram of Aquanaut is shown in Figure C.1. The main Aquanaut interface consists of an OpenGL display and menu options for loading raw data. Aquanaut operates in two modes, the *simulation* mode (see Figure C.3(a)) and the *analyser* mode (see Figure C.3(b)).

In the simulation mode, basic geometric primitives can be added to a sonar simulator scene. The simulator can produce sonar beam intersections with the primitives. These intersections can be used as starting points for reconstruction using the available homotopy based methods. Available options in this mode are

- Dummy sonar parameters such as the type of sonar (stack of 2D frames or 3D volume) with its parameters (such as vertical/horizontal angle, number of beams per frame, and beam length),
- Adding scene objects, loading a scene, and saving an existing scene,
- Modifying spatial properties of an object in the scene. An object in the scene can be translated, rotated and scaled arbitrarily in the scene, and

- Reconstruction of the existing scene. The simulation tab provides options to perform the beam-object intersection, and using those intersections, perform a reconstruction using the available homotopy methods. A reconstruction can be saved either as an implicit homotopy field (a VTK volume) or as an iso-surface mesh (OBJ format).

The analyser mode, on the other hand, works on the real data. Signal masses can be read into Aquanaut and these serve as beam-object intersections. Similar options to save the reconstruction are available here as well.

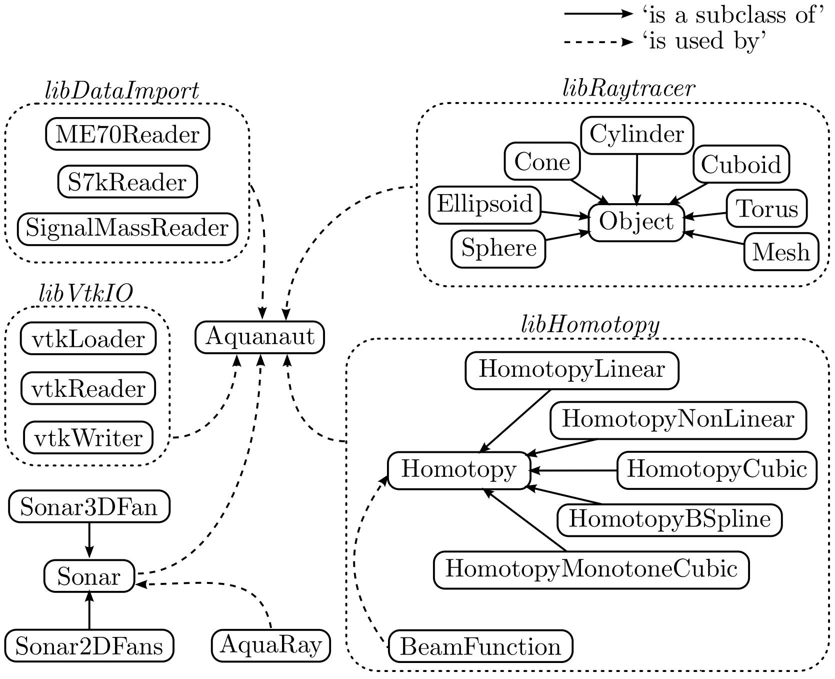


Figure C.1: *Aquanaut class diagram.*

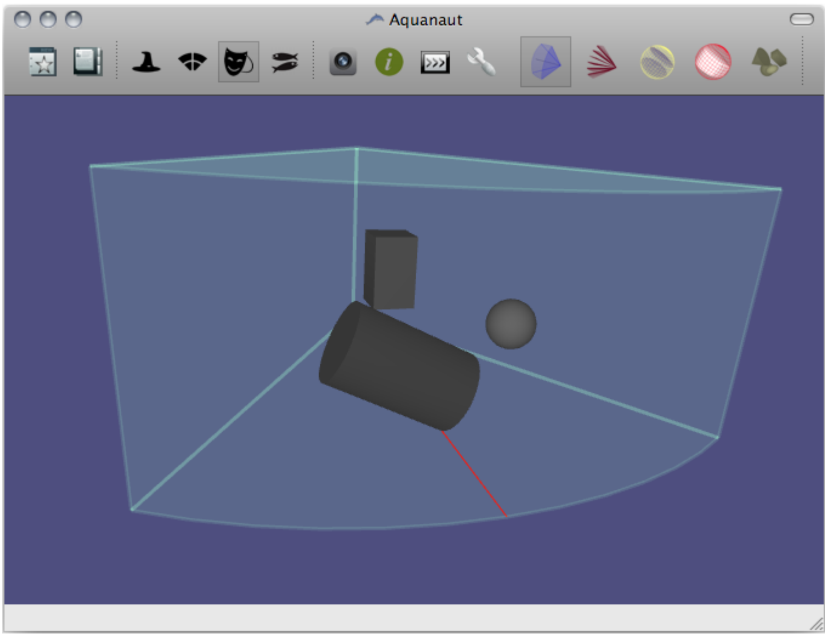
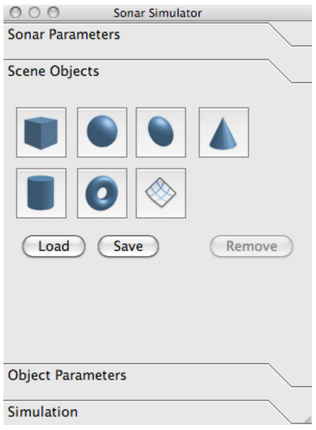
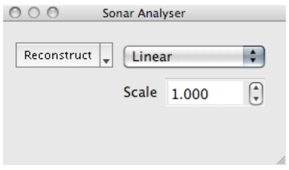


Figure C.2: *Aquanaut main window.*



(a) Simulator dialog



Reconstruct Linear

Scale 1.000

(b) Analyser dialog

Figure C.3: *Aquanaut modes of operation.*

Bibliography

[1] Lindem Data Acquisition. Sonar4 and Sonar5-Pro post processing systems, 2010. http://www.fys.uio.no/~hbalk/sonar4_5/Downloads.htm.

[2] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gärtner. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995.

[3] E. L. Allgower and K. Georg. *Numerical continuation methods: an introduction.* Springer-Verlag New York, Inc. New York, NY, USA, 1990.

[4] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, 22(4):481–504, 1999.

[5] M. A. Armstrong. *Basic topology.* Springer, 1983.

[6] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. Mesh: Measuring errors between surfaces using the Hausdorff distance. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, volume 1, pages 705–708. Citeseer, 2002.

[7] J. A. Bærentzen. Introduction to GEL, 2010. http://www2.imm.dtu.dk/projects/GEL/.

[8] X. Bai, L. J. Latecki, and W. Y. Liu. Skeleton pruning by contour partitioning with discrete curve evolution. *IEEE transactions on pattern analysis and machine intelligence*, 29(3):449–462, 2007.

[9] C. Baillard, C. Barillot, and P. Bouthemy. Robust adaptive segmentation of 3D medical images with level sets. *Rapport de recherche-Institut National de Recherche en Informatique et en Automatique*, 2000.

[10] C. Bajaj, G. Xu, and Q. Zhang. A higher order level set method with applications to smooth surface constructions. ICES report 06-18. *Institute for Computational Engineering and Sciences, The University of Texas at Austin*, 2006.

[11] C. Bajaj, G. Xu, and Q. Zhang. Smooth surface constructions via a higher-order level-set method. *Pro. CAD/GRAPHICS*, 2007.

[12] C. L. Bajaj, E. J. Coyle, and K. N. Lin. Arbitrary topology shape reconstruction from planar cross sections. *Graphical models and image processing*, 58(6):524–543, 1996.

[13] C. L. Bajaj, G. Xu, and Q. Zhang. A fast variational method for the construction of resolution adaptive C^2-smooth molecular surfaces. *Computer Methods in Applied Mechanics and Engineering*, 198(21-26):1684–1690, 2009.

[14] C. L. Bajaj, G. L. Xu, and Q. Zhang. Higher-order level-set method and its application in biomolecular surfaces construction. *Journal of Computer Science and Technology*, 23(6):1026–1036, 2008.

[15] J. P. Balabanian, I. Viola, E. Ona, R. Patel, and M. E. Gröller. Sonar explorer: A new tool for visualization of fish schools from 3D sonar data. In *Data Visualization - EuroVis 2007*, pages 155–162. IEEE, 5 2007.

[16] I. Beros and M. Marusic. Evaluation of tension splines. *Mathematical Communications*, 4:73–81, 1999.

[17] D. Berzin and I. Hagiwara. Minimal area for surface reconstruction from cross sections. *The Visual Computer*, 18(7):437–444, 2002.

[18] G. E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, November 1990.

[19] J. D. Boissonnat. Shape reconstruction from planar cross sections. *Computer vision, graphics, and image processing*, 44(1):1–29, 1988.

[20] J. D. Boissonnat and P. Memari. Shape reconstruction from unorganized cross-sections. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, page 98. Eurographics Association, 2007.

[21] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, 1986.

[22] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, page 76. ACM.

[23] U. Castellani, M. Cristani, and V. Murino. Acoustic range image segmentation by effective mean shift. In *2006 IEEE International Conference on Image Processing*, pages 2437–2440, 2006.

[24] CGAL. Computational Geometry Algorithms Library, 2010. `http://www.cgal.org`.

[25] T. F. Chan and L. Vese. A level set algorithm for minimizing the Mumford-Shah functional in image processing. *IEEE/Computer Society Proceedings of the 1st IEEE Workshop on Variational and Level Set Methods in Computer Vision*, pages 161–168, 2001.

[26] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2), 2001.

[27] T. F. Chan and L. A. Vese. A level set algorithm for minimizing the mumford-shah functional in image processing. In *IEEE Workshop on Variational and Level Set Methods*, pages 161–168, 2001.

[28] B. Chapman, G. Jost, R. Van der Pas, and D. J. Kuck. *Using OpenMP: portable shared memory parallel programming.* The MIT Press, 2007.

[29] Y. Chen and A. Raheja. Wavelet lifting for speckle noise reduction in ultrasound images. *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pages 3129–3132, 2005.

[30] D. Comaniciu and P. Meer. Robust analysis of feature spaces: Color image segmentation. In *1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997. Proceedings.*, pages 750–755, 1997.

[31] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[32] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of symbolic computation*, 9(3):251–280, 1990.

[33] P. Costantini and R. Morandi. Monotone and convex cubic spline interpolation. *Calcolo*, 21(3):281–294, 1984.

[34] M. D. Crossley. *Essential topology.* Springer Verlag, 2005.

[35] L. Dagum and R. Menon. OpenMP: an industry standard API for shared-memory programming. *IEEE Computational Science & Engineering*, 5(1):46–55, 1998.

[36] D. F. Davidenko. On a new method of numerical solution of systems of nonlinear equations. In *Dokl. Akad. Nauk SSSR*, volume 88, 1953.

[37] C. de Boor and B. Swartz. Piecewise monotone interpolation. *Journal of Approximation Theory*, 21(4):411–416, 1977.

[38] C. Den Heijer and W. C. Rheinboldt. On steplength algorithms for a class of continuation methods. *SIAM Journal on Numerical Analysis*, 18(5):925–948, 1981.

[39] H. Derin and H. Elliott. Modeling and segmentation of noisy and textured images using Gibbs random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):39–55, 1987.

[40] K. Djemal. Speckle reduction in ultrasound images by minimization of total variation. In *IEEE International Conference on Image Processing, 2005. ICIP 2005*, volume 3, 2005.

[41] D. L. Donoho. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3):613–627, 1995.

[42] G. Dougherty and J. Varro. A quantitative index for the measurement of the tortuosity of blood vessels. *Medical Engineering & Physics*, 22(8):567–574, 2000.

[43] J. Dunlop. Statistical modelling of sidescan sonar images. *OCEANS'97. MTS/IEEE Conference Proceedings*, 1, 1997.

[44] V. Dutt and J. F. Greenleaf. Adaptive speckle reduction filter for log-compressed B-scan images. *Medical Imaging, IEEE Transactions on*, 15(6):802–813, 1996.

[45] Message Passing Interface Forum. MPI: A message-passing interface standard, 1994.

[46] V. S. Frost, J. A. Stiles, K. S. Shanmugan, and J. C. Holtzman. A model for radar images and its application to adaptive digital filtering of multiplicative noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:157–166, 1982.

[47] H. Fuchs, Z. M. Kedem, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, 1977.

[48] K. Fujimura and E. Kuo. Shape reconstruction from contours using isotopic deformation. *Graphical Models and Image Processing*, 61(3):127–147, 1999.

[49] L. Gagnon. Wavelet filtering of speckle noise-some numerical results. *Proc. on the Conference Vision Interface ('99), Trois-Rivieres, Canada*, pages 1–8, 1999.

[50] K. Georg. Numerical integration of the Davidenko equation. *Numerical solution of nonlinear equations*, pages 128–161, 1981.

[51] C. Gold, D. Mioc, F. Anton, O. Sharma, and M. Dakowicz. A methodology for automated cartographic data input, drawing and editing using kinetic Delaunay/Voronoi diagrams. *Generalized Voronoi Diagram: A Geometry-Based Approach to Computational Intelligence*, pages 159–196, 2008.

[52] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Prentice Hall, 2007.

[53] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Procesisng.* Prentice Hall, 2 edition, 2002.

[54] N. K. Govindaraju, B. Lloyd, W. Wang, M. Lin, and D. Manocha. Fast computation of database operations using graphics processors. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 206, New York, NY, USA, 2005. ACM.

[55] A. Grama, A. Gupta, G. Karypis, and V. Kumar. *Introduction to parallel computing.* Addison Wesley, 2003.

[56] T. Gunji, S. Kim, M. Kojima, A. Takeda, K. Fujisawa, and T. Mizutani. PHoM–a polyhedral homotopy continuation method for polynomial systems. *Computing*, 73(1):57–77, 2004.

[57] J. L. Gustafson. Reevaluating Amdahl's law. *Communications of the ACM*, 31:532–533, 1988.

[58] G. Haines. *Sound underwater.* David & Charles, 1974.

[59] X. Hao, S. Gao, and X. Gao. A novel multiscale nonlinear thresholding method for ultrasonic speckle suppressing. *IEEE Transactions on Medical Imaging*, 18(9):787–794, 1999.

[60] M. Harris, S. Sengupta, and J. D. Owens. Parallel prefix sum (scan) with CUDA. *GPU Gems*, 3, 2007.

[61] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics (ACM)*, 26(2):71–78, 1992.

[62] J. M. Hyman. Accurate monotonicity preserving cubic interpolation. *SIAM journal on scientific and statistical computing*, 4(4):645–654, 1983.

[63] L3 Communications SeaBeam Instruments. How mapping sonars work, 2010. `http://www.ldeo.columbia.edu/res/pi/MB-System/sonarfunction/SeaBeamMultibeamTheoryOperation.pdf`.

[64] A. Isar, D. Isar, S. Moga, J. M. Augustin, and X. Lurton. Multi-scale MAP despeckling of sonar images. *Oceans 2005-Europe*, 2, 2005.

[65] I. M. James. *History of topology.* North-Holland, 1999.

[66] K. Jänich. Topology, undergraduate texts in mathematics, 1984.

[67] J. P. Keener and H. B. Keller. Perturbed bifurcation theory. *Archive for Rational Mechanics and Analysis*, 50(3):159–175, 1974.

[68] H. B. Keller. Nonlinear bifurcation. *Journal of Differential Equations*, 7(3):417–434, 1970.

[69] H. B. Keller. Numerical solution of bifurcation and non linear eigenvalue problems. In *Applications of Bifurcation Theory: Proceedings of an Advanced Seminar*, pages 359–384, 1977.

[70] E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research and Development*, 19(1):2–11, 1975.

[71] R. R. Kneipfer. Sonar beamforming-an overview of its history and status. Technical Report A981052, Naval Undersea Warfare Center Detachment, Connecticut, April 1992.

[72] R. J. Korneliussen. Measurement and removal of echo integration noise. *ICES Journal of Marine Science*, 57(4):1204, 2000.

[73] K. Krissian, K. Vosburgh, R. Kikinis, and C. F. Westin. Anisotropic diffusion of ultrasound constrained by speckle noise model. *Department of Radiology, Brigham and Women's Hospital, Harvard Medical School, Laboratory of Mathematics in Imaging, Tech. Rep*, 4, 2004.

[74] K. Krissian and C.F. Westin. Fast sub-voxel re-initialization of the distance map for level set methods. *Pattern Recognition Letters*, 26(10):1532–1542, 2005.

[75] D. T. Kuan, A. A. Sawchuk, T. C. Strand, and P. Chavel. Adaptive noise smoothing filter for images with signal-dependent noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(2):165–177, 1985.

[76] M. Kubíček. Dependence of solution of nonlinear systems on a parameter. *ACM Transactions on Mathematical Software*, 2(1):107, 1976.

[77] L. J. Latecki and R. Lakamper. Polygon evolution by vertex deletion. In *Proceedings of the Second International Conference on Scale-Space Theories in Computer Vision*, pages 398–409. Springer, 1999.

[78] J. S. Lee. Refined filtering of image noise using local statistics. *Computer graphics and image processing*, 15(4):380–389, 1981.

[79] A. E. Lefohn, J. M. Kniss, C. D. Hansen, and R. T. Whitaker. A streaming narrow-band algorithm: Interactive computation and visualization of level sets. *IEEE Transactions on Visualization and Computer Graphics*, 10:422–433, 2004.

[80] A. E. Lefohn, J. M. Kniss, C. D. Hansen, and R. T. Whitaker. A streaming narrow-band algorithm: interactive computation and visualization of level sets. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):422–433, 2004.

[81] J. Li, C. A. Papachristou, and R. Shekhar. A “brick” caching scheme for 3D medical imaging. In *ISBI*, pages 563–566, 2004.

[82] M. Lianantonakis and Y. R. Petillot. Sidescan sonar segmentation using active contours and level set methods. *Oceans 2005-Europe*, 1, 2005.

[83] L. Liu, C. Bajaj, J. O. Deasy, D. A. Low, and T. Ju. Surface reconstruction from non-parallel curve networks. In *Computer Graphics Forum*, volume 27, page 155. Citeseer, 2008.

[84] A. Lopes, R. Touzi, and E. Nezry. Adaptive speckle filters and scene heterogeneity. *IEEE Transactions on Geoscience and Remote Sensing*, 28(6):992–1000, 1990.

[85] T. Loupas, W. N. McDicken, and P. L. Allan. An adaptive weighted median filter for speckle suppression in medical ultrasonic images. *Circuits and Systems, IEEE Transactions on*, 36(1):129–135, 1989.

[86] X. Lurton. *An introduction to underwater acoustics: principles and applications.* Springer-Praxis, 2002.

[87] R. Malladi and J. A. Sethian. Image processing via level set curvature flow. *Proceedings of the National Academy of Sciences of the United States of America*, 92(15):7046–7050, 1995.

[88] R. Malladi, J.A. Sethian, and B.C. Vemuri. A topology independent shape modeling scheme. In *SPIE Proceedings on Geometric Methods in Computer Vision II, San Diego*, pages 246–258, 1993.

[89] P. Memari and J. D. Boissonnat. Provably good 2D shape reconstruction from unorganized cross-sections. In *Computer Graphics Forum*, volume 27, pages 1403–1410, 2008.

[90] M. Meyer, H. Lee, A. Barr, and M. Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of Graphics Tools*, 7(1):13–22, 2002.

[91] M. Mignotte, C. Collet, P. Perez, and P. Bouthemy. Unsupervised Markovian segmentation of sonar images. *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, 4, 1997.

[92] M. Mignotte, C. Collet, P. Perez, and P. Bouthemy. Sonar image segmentation using an unsupervised hierarchical MRF model. *IEEE Transactions on Image Processing*, 9(7):1216–1231, 2000.

[93] R. E. Moore and M. J. Cloud. *Computational functional analysis.* Horwood Pub Ltd, 2007.

[94] A. P. Morgan, A. J. Sommese, and C. W. Wampler. A product-decomposition bound for Bezout numbers. *SIAM Journal on Numerical Analysis*, 32(4):1308–1325, 1995.

[95] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Commun. Pure Appl. Math.*, 42(5):577–685, 1989.

[96] J. Munkres. *Topology (2nd Edition).* Prentice Hall, 1999.

[97] Nvidia. Compute Unified Device Architecture-programming guide version 2.0.

[98] Nordsøen Oceanarium. North Sea Oceanarium, 2010. `http://www.nscentre.dk/Default.aspx?ID=1895`.

[99] E. Ona, L. N. Andersen, H. P. Knudsen, and S. Berg. Calibrating multibeam, wideband sonar with reference targets. *OCEANS 2007-Europe*, pages 1–5, 2007.

[100] E. Ona, V. Mazauric, and L. N. Andersen. Calibration methods for two scientific multibeam systems. *ICES Journal of Marine Science*, 66(6):1326, 2009.

[101] S. Osher and R. P. Fedkiw. *Level sets and dynamic implicit surfaces.* Springer New York, 2003.

[102] S. Osher and L. I. Rudin. Feature-oriented image enhancement using shock filters. *SIAM Journal on Numerical Analysis*, 27(4):919–940, 1990.

[103] S. J. Osher and J. A. Sethian. Fronts propagation with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[104] G. Papandreou and P. Maragos. Multigrid geometric active contour models. *IEEE Transactions on Image Processing*, 16(1):229, 2007.

[105] T. N. Pappas. An adaptive clustering algorithm for image segmentation. *IEEE Transactions on Signal Processing*, 40(4):901–914, 1992.

[106] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[107] S. Pruess. An algorithm for computing smoothing splines in tension. *Computing*, 19(4):365–373, 1978.

[108] I. Quidu, J. P. Malkasse, G. Burel, and P. Vilbé. A 2-D filter specification for sonar image thresholding. *Advanced Concepts for Intelligent Vision Systems (ACIVS'2001) conference, Baden-Baden, Germany*, 2001.

[109] L. B. Rall. Davidenko's method for the solution of nonlinear operator equations. *MRC Technical Summary Report*, 948, 1968.

[110] RESON. SeaBat Multibeam Sonars - Product Range, 2010. `http://www.reson.com/sw1122.asp`.

[111] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[112] P. K. Sahoo, S. Soltani, and A. K. C. Wong. A survey of thresholding techniques. *Computer vision, graphics, and image processing*, 41(2):233–260, 1988.

[113] J. A. Sethian. Theory, algorithms, and applications of level set methods for propagating interfaces. *Acta Numerica 1996*, pages 309–395, 1996.

[114] R. Seydel. Numerical computation of branch points in nonlinear equations. *Numerische Mathematik*, 33(3):339–352, 1979.

[115] O. Sharma and F. Anton. CUDA based level set method for 3D reconstruction of fishes from large acoustic data. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 17*, 2009.

[116] O. Sharma and F. Anton. Homotopic object reconstruction using natural neighbor barycentric coordinates. In *Seventh International Symposium on Voronoi Diagrams in Science and Engineering (ISVD)*, 2010.

[117] O. Sharma and F. Anton. Homotopy based surface reconstruction with application to acoustic signals. *The Visual Computer Journal*, accepted for publication in 2010.

[118] O. Sharma and F. Anton. Homotopy based 2D topologic reconstruction from arbitrary linear cross sections. *The Visual Computer Journal*, submitted for publication in 2010.

[119] O. Sharma, F. Anton, and D. Mioc. On the isomorphism between the medial axis and a dual of the Delaunay graph. In *2009 Sixth International Symposium on Voronoi Diagrams*, pages 89–95. IEEE, 2009.

[120] O. Sharma, D. Mioc, and F. Anton. Feature extraction and simplification from colour images based on colour image segmentation and skeletonization using the Quad-edge data structure. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 15*, pages 225–232, 2007.

[121] O. Sharma, D. Mioc, and F. Anton. Polygon feature extraction from satellite imagery based on color image segmentation and medial axis. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37:235–240, 2008.

[122] O. Sharma, Q. Zhang, F. Anton, and C. Bajaj. Multi-domain, higher order level set scheme for 3D image segmentation on the GPU. In *23rd IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[123] O. Sharma, Q. Zhang, F. Anton, and C. Bajaj. Fast, streaming 3D levelset on the GPU for smooth multi-phase segmentation. *LNCS Transactions on Computational Science*, submitted for publication in 2010.

[124] Y. Shinagawa and T. L. Kunii. The homotopy model: a generalized model for smooth surface generation from cross sectional data. *The Visual Computer*, 7(2):72–86, 1991.

[125] A. Sidlesky, G. Barequet, and C. Gotsman. Polygon reconstruction from line cross-sections. In *Canadian Conference on Computational Geometry*, 2006.

[126] E. J. Simmonds and D. N. MacLennan. *Fisheries Acoustics: Theory and Practice*. Blackwell Publishing, 2005.

[127] Simrad. Simrad EY500 instruction manual, 2010. `http://www.simrad.com/www/01/NOKBG0397.nsf/AllWeb/E3B2DE7D58D2526DC125711E003E9E91/$file/130078ag_ey500_instruction_manual_english.pdf?OpenElement`.

[128] Simrad. Simrad MS70 Scientific multibeam sonar, 2010. `http://www.simrad.com/www/01/NOKBG0240.nsf/AllWeb/99256F52D98D2ED2C125719C002C487F?OpenDocument`.

[129] P. Soille. *Morphological image analysis: principles and applications.* Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2003.

[130] A. J. Sommese and C. W. Wampler. *The Numerical solution of systems of polynomials arising in engineering and science.* World Scientific Publishing Company, 2005.

[131] H. Späth. Exponential spline interpolation. *Computing*, 4(3):225–233, 1969.

[132] R. Steffens and T. Theobald. Some new techniques for explicit Mixed Volume computation. In *Proceedings of the Rhine Workshop on Computer Algebra*, 2008.

[133] R. Strzodka and M. Rumpf. Level set segmentation in graphics hardware. In *Proc. IEEE International Conference on Image Processing*, pages 1103–1106, 2001.

[134] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40:1528–1538, 1995.

[135] UT-CVC. Volume Rover, 2006. `http://cvcweb.ices.utexas.edu/cvc/projects/project.php?proID=9`.

[136] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software*, 25(2):251–276, 1999.

[137] L. A. Vese and T. F. Chan. A multiphase level set framework for image segmentation using the Mumford and Shah model. *International Journal of Computer Vision*, 50(3):271–293, 2002.

[138] E. L. Wachspress. *A rational finite element basis.* Academic Press, 1975.

[139] J. L. Watkins and A. S. Brierley. A post-processing technique to remove background noise from echo integration data. *ICES Journal of Marine Science*, 53(2):339, 1996.

[140] L. T. Watson. Numerical linear algebra aspects of globally convergent homotopy methods. *SIAM Review*, 28(4):529–545, 1986.

[141] L. T. Watson, S. C. Billups, and A. P. Morgan. Algorithm 652: HOMPACK: A suite of codes for globally convergent homotopy algorithms. *ACM Transactions on Mathematical Software*, 13(3):310, 1987.

[142] T. L. Wayburn and J. D. Seader. Homotopy continuation methods for computer-aided process design. *Computers & Chemical Engineering*, 11(1):7–25, 1987.

[143] J. Weickert, B. Romeny, and M. A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Transactions on Image Processing*, 7(3):398–410, 1998.

[144] G. Wolberg and I. Alfy. Monotonic cubic spline interpolation. In *Computer Graphics International, 1999. Proceedings*, pages 188–195, 1999.

[145] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo. Weighted median filters: a tutorial. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 43(3):157–192, 1996.

[146] Y. Zhang, R. Rohling, and D. K. Pai. Direct surface extraction from 3D freehand ultrasound images. In *IEEE Visualization*, pages 45–52, 2002.

Index