Interactive Measurements of Three-Dimensional Branching Objects

Copyright © 2009 by Krzysztof Jakub Kruszyński.

All rights reserved. No part of this book may be reproduced, stored in a database or retrieval system, or published, in any form or in any way, electronically, mechanically, by print, photoprint, microfilm or any other means without prior written permission of the author.

ISBN: 90 6196 556 X
ISBN-13: 978 90 6196 556 5

This work was carried out in the context of the Virtual Laboratory for e-Science project (`http://www.vl-e.nl`). This project is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation program of the Ministry of Economic Affairs (EZ).

Interactive Measurements of Three-Dimensional Branching Objects

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op maandag 25 januari 2010 om 16.00 uur

door

Krzysztof Jakub Kruszyński

geboren te Warschau, Polen

Dit proefschrift is goedgekeurd door de promotor:

prof.dr.ir. R. van Liere

DLA KATARZYNY

Table of Contents

Preface ... **iii**

Chapter 1 Introduction ... **1**

1.1 Measuring Coral ... 2
1.2 Approach ... 4
1.3 Research Goals ... 5
1.4 Structure ... 6
1.5 Publications ... 6

Chapter 2 Related Work/Survey ... **9**

2.1 Systems ... 9
2.2 Overview ... 12

Chapter 3 Volume to Skeleton ... **15**

3.1 Noise Filtering ... 15
3.2 Segmentation ... 19
3.3 Skeletonization ... 24
3.4 Choosing algorithms experimentally ... 26
3.5 Conclusion ... 36

Chapter 4 Measuring the Skeleton ... **39**

4.1 Skeleton Graph ... 39
4.2 Metrics ... 41
4.3 Accuracy and Uncertainty ... 44
4.4 Application ... 46
4.5 Conclusion ... 53

Chapter 5 Visualization of the Measurements ... **55**

5.1 Editing ... 55
5.2 Exploring Measurements ... 57
5.3 Uncertainty Visualization ... 63
5.4 Conclusion ... 73

Chapter 6 Tangible Interaction ... **75**

6.1 Introduction ... 75
6.2 Related Work ... 76
6.3 Tangible Prop Concept ... 78
6.4 Motivation ... 81

6.5 Tangible Interaction Techniques 88
6.6 Integration with Augmented Reality 97
6.7 Tangible Corals 99
6.8 Conclusion 104

Chapter 7 Conclusion and Future Work 105
7.1 Future work 106

Appendix A 3D input in VTK 109
A.1 Input in VTK 109
A.2 VRPN 110
A.3 Our Approach 110
A.4 3D-aware VTK Widgets 114

Bibliography 117

List of Figures 123

Summary 125

Samenvatting 127

Streszczenie 129

Curriculum Vitae 131

Preface

The work described in this thesis was performed in the Visualization and 3D Interfaces research group (INS3) of the Centrum Wiskunde & Informatica (CWI) in Amsterdam, in the period from 2004 to 2009.

During this time there were many people who have helped and supported me, and I would like to take a brief moment to thank them here.

My promotor and supervisor Robert van Liere, for his many insights, fascinating discussions, and for making this work possible in the first place.

Jaap Kaandorp, for providing the data, methods, and background information concerning corals, and for all his help and enthusiasm.

My colleagues from INS3 – Alexander, Arjen, Bregt, Ferdi, Lei, Si, Wojciech – for many interesting insights and stories, and all the other colleagues at CWI for creating an enjoyable work environment, and everybody else whom I had the pleasure meeting in the course of my work.

The other members of the reading committee, Erik Janssen, Bart ter Haar Romeny and Jack van Wijk, for their scrupulous reading and invaluable comments.

My parents, for giving me care and inspiration.

My family and friends, for their support and interest.

Finally, and most importantly, I thank my wife Katarzyna, for her love, support, understanding, patience, and for simply being there. I could not have done this without you.

Krzysztof Kruszyński
Almere, December 2009

Chapter 1 Introduction

A basic way to learn more about geometric objects is to measure them. Casual investigation of geometric objects will yield only qualitative information about the object. To truly gain understanding it is necessary to obtain quantitative measurement data that can easily be analyzed and compared. To do so by hand may be feasible for simple objects, but as the complexity of an object increases the task becomes more and more laborious, if not nearly impossible, up to a point where most of the time is spent on obtaining data rather than interpreting it and gaining new knowledge.

Computers aid researchers in the analysis of large amounts of data. However, the ability to process increasing amounts of data leads to the desire to acquire even larger amounts of data for analysis. This leads to the need to involve automated systems not only for the analysis but also for acquiring the data.

Obtaining large amounts of measurements of complex objects is a common task in many areas of research and engineering. A much used method to measure objects is to acquire a digital version of the object, and to perform the measurements on this virtual replica. Quantitative measurement data can be obtained directly or indirectly. Directly by probing the digitized values, or by searching for patterns in them. It can also be done indirectly, by generating or extracting an alternative, geometric model of the data, for example an iso-surface or a skeleton. This alternative representation can then be measured, or it can be used as a guide to analyze the original data.

A digitized version of an object will only in rare cases contain the actual relevant measurements. Digitization either yields a large number of samples of the location of the surface of an object, or a map (image) of one or more properties of some enclosed area that includes at least part of the object, such as the three-dimensional density map that is produced by a CT scanner. While such data is useful for (interactive) visual analysis, to measure the object it is necessary to locate the relevant features of the object in the data, which involves converting the data to a suitable intermediate representation. The

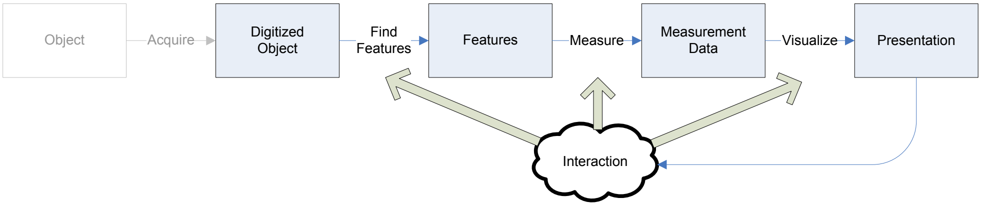


Figure 1.1: Global overview of the measurement pipeline. An object is acquired, features are located in the data, and the located features are measured.

measurement process is a pipeline involving a number of steps, a global overview of which is shown in Figure 1.1.

1.1 Measuring Coral

Our goal is to create an interactive measurement environment for measuring branching objects, specifically marine branching stony corals. These corals are highly complex branching tree-like structures [42]. The shapes of the corals are of great interest to marine biologists. The shape of the colonies of many species is much more dependent on environmental factors than on genetics, and thus can be a source of information on current and, due to the longevity and slow growth rate of coral colonies, also on past environmental parameters. By analyzing the shape of colonies, biologists can determine how the climate has changed.

Computational models have been created that can simulate the growth of coral colonies, and the response to various environmental factors [52]. The validation of these models occurs through analyzing the similarity of the resulting shapes to the shapes of actual coral colonies of a particular species. This requires a quantitative analysis of the morphological traits of both the simulated and the actual colonies.

Traditionally corals have been analyzed and described in qualitative terms. Quantitative data is obtained by performing measurements on specimens, or on photographs of specimens, using a manual process. This greatly limits the amount of data that can be obtained from a specimen, and it limits the accuracy of the measurements.

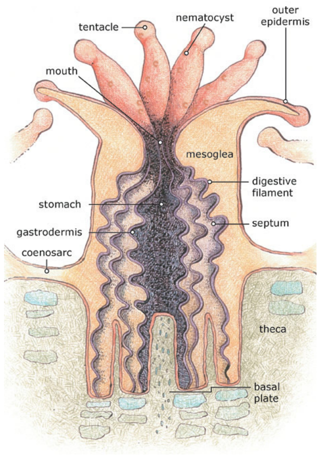


Figure 1.2: A coral polyp [55].

1.1.1 Coral Anatomy

Scleractinia, or stony corals, are an order of small marine animals belonging to the phylum *Cnidaria*, class *Anthozoa*. Although some species are solitary, many form large colonies of individual, but interconnected polyps. The individual polyps secrete external skeletons called corallites, which are light, porous structures made of calcium carbonate. A single corallite is shaped like a cup with radial plates (septa) inside, with the coral polyp protruding from the top. As more skeletal tissue is secreted, the corallites become longer. In colony-forming corals the polyps reproduce through asexual division, remaining connected with each other through the secreted skeleton and through connecting soft tissue. A cross-section of a polyp can be seen in Figure 1.2.

Although the polyps of colonial corals are at most a few millimeters across, the colonies themselves can be huge, as the polyps continue to divide and secrete new skeletons. While the shape of both the polyp and its skeleton is fixed, the shape of the colony can vary significantly even within a single species. Some colonies are flat, others are spherical, and yet others assume branching tree-like shapes. After a colony dies, the skeleton remains and is often reused as a firm attachment point for other creatures, possibly even other corals. The skeletons of these colonies are thus the basic building blocks of coral reefs. It is also the morphology of these skeletons that is studied.

The skeletons of branching corals, although geometrically complex, appear to be relatively simple objects with a smooth curved surface. However, this is not the case. The coral possesses the following characteristics:

- The surface has a rough texture and is uneven;
- The material is not homogenous; there are different types of structures, and there is variation within these structures as well;
- The coral contains various damage artifacts;
- Remains of other organisms are present on the coral.

1.1.2 Data Acquisition

The data used in this work is acquired using a CT scanner. These machines have a number of parameters, which are set by the operator to obtain optimal image quality. The acquisition of CT scans of coral colonies has not yet been practiced on a large scale, it is therefore conceivable that the CT scan parameters that are used are not optimal for coral, thus leading for example to more noise being present in the images than might otherwise be possible.

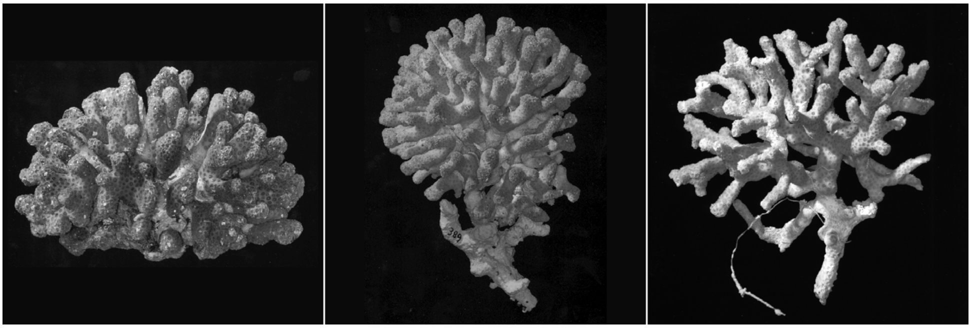


Figure 1.3: Photographs of the three coral specimens.

CT scans of corals are characterized by the following concerns:

- The scans contain noise;
- Fine porous structures show up as low density material in the images due to limited resolution;
- The resolution of the scanner also affects the apparent topology, creating artificial fusion of very tightly spaced branches.

These characteristics present significant difficulties for extraction algorithms, and hence for obtaining robust measurements of the coral. Some of these difficulties have no automatic solution, and require interaction to solve.

A large part of the research in this thesis makes use of three scanned specimens of the *Madracis mirabilis* coral, commonly called yellow pencil coral due to its color and branch diameter. This species is commonly found in the Caribbean, where colonies of over 5m in diameter have been encountered. The size of individual polyps of this species is between 1.1mm and 1.6mm. Photographs of the three specimens are shown in Figure 1.3. These colonies have a diameter of 10cm to 15cm, and were collected at depths between 6m (left photo) and 20m (right photo) at the reef of Curaçao (Netherlands Antilles, 12^0 N 69^0 W). The colonies were cleaned of the polyps and the skeletons were transported to Amsterdam. Three-dimensional images of the colonies were obtained using a medical scanner (Philips Mx8000). The three coral colonies were scanned at a (nearly isotropic) resolution of 0.25 mm × 0.25 mm × 0.3 mm, with a slice size of 512 × 512 pixels. An isotropic resolution is obtained by using equal slice spacings in all dimensions. Due to technical limitations in medical scanners, anisotropic resolutions are frequently used and a lower resolution is applied in one dimension, for example in [41] a slice thickness of 2.5 mm was used. In the reconstruction process the higher resolution in the other dimensions is used to approximate the object. By applying a nearly isotropic resolution this approximation could be avoided. The numbers of slices in the samples were 329, 541 and 373 in each scan respectively.

1.2 Approach

The approach uses Computed Tomography (CT) scans of the corals. From these scans we extract the morphological skeleton of the coral, which we then measure using a variety of relevant morphological metrics which are used by coral biologists [40,41]. The resulting data can then be used for statistical analysis of the shape of a specimen, or for a quantitative comparison between different specimens. The morphological skeletons of the colonies shown in Figure 1.3 are shown in Figure 1.4. Approximately 350 branches were detected in the left and middle specimen, and approximately 130 in the specimen on the right.

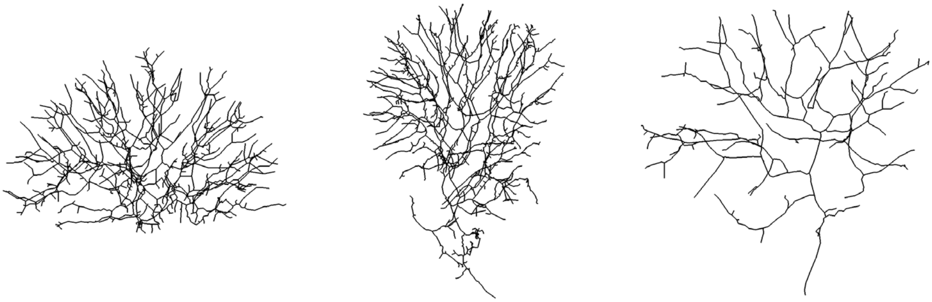


Figure 1.4: Skeletons of the corals in Figure 1.3.

An important issue is the existence and propagation of uncertainty in the data, and the sensitivity of algorithms to noise. We use an objective quantitative method to select a suitable skeletonization algorithm, and our visualizations show the uncertainty in the data.

Interaction plays an important role in the measuring environment. On one hand it is used to assist in the extraction of the data, where artifacts in the coral are analyzed and corrected, while on the other hand it is used to interactively explore and analyze the result data. Advanced tangible interfaces, using three-dimensional printed coral replicas, make the interaction easy and natural.

1.2.1 Coral Model

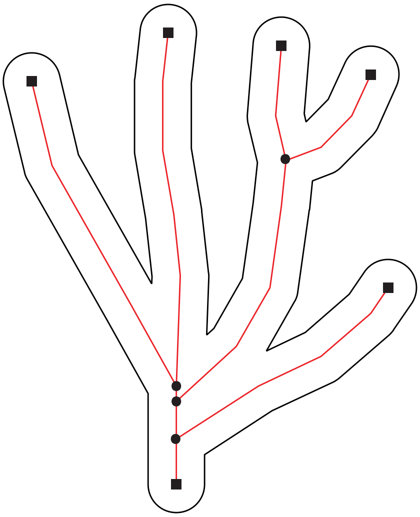


Figure 1.5: The model of a coral colony.

We model a coral as a simple graph G with vertices V and edges E. Each vertex is associated with coordinates in 3D space, thus each edge defines a line segment in 3D space.

We also define the branches $B(G)$. A branch b of a coral graph G is a path in the graph G between two vertices v_0 and v_n where the degree is

$$d_G(v_0) \neq 2$$
$$d_G(v_n) \neq 2$$
$$d_G(v_{1\dots n-1}) = 2.$$

Also, the path $v_0 v_n$ identifies the same branch as the opposite path $v_n v_0$.

A terminal branch is a branch that contains a vertex v with $d_G(v) = 1$. A junction, or branching point, is a vertex v with $d_G(v) > 2$.

An offset surface is associated with this graph. The distance function $D_S(v)$ gives the distance of the surface at each vertex of the graph. This surface corresponds to the surface of an actual coral colony.

This coral model is shown in Figure 1.5. Vertices with $d_G(v) \neq 2$ are indicated with square and round symbols. The branches B can be seen as the line segments between these symbols. The round symbols indicate the branching points ($d_G(v) > 2$), the square symbols indicate the endpoints of terminal branches ($d_G(v) = 1$). The 3D surface can be seen as the outline surrounding this graph.

On this model a number of biologically relevant morphological metrics are defined. These are described in section 4.2.1.

1.3 Research Goals

The subject matter of the research is interactive measurement of three-dimensional branching structures, with a particular focus on branching marine corals. The morphology of corals has never previously been analyzed in this manner; it is the first time that coral morphology has been analyzed using CT scans and 3D morphological skeletons. The measuring system that has been developed in the course of this work is currently being used by coral researchers.

Three questions are addressed in this research:

- What is the effect of the sensitivity of skeleton extraction algorithms in relation to robust and accurate measurements of branching objects?
- What is the role of 3D interaction and automation in the measurement process?
- What is the role of augmented reality and tangible interfaces for interactive shape analysis of branching objects?

Much of the work is not exclusive to branching corals, but has applicability to other similar objects. Skeletons are used to measure other branching tube-like structures, for example the vascular system.

1.4 Structure

The thesis is structured as follows. Chapter 2 places this work in the context of other measuring environments. Chapter 3 describes the processing steps leading from a scanned volume to a skeleton. Also, a skeletonization algorithm is selected based on its performance on coral-like structures. Chapter 4 treats the measuring of the coral. This includes the difficulties in obtaining the measurements, the implementation of the measurements, and the results of measuring three specimens. Chapter 5 deals with interaction and visualization. Highly interactive 2D and 3D visualizations are used to explore the measurements. Also, the visualization of uncertainty is investigated. Chapter 6 is concerned with tangible 3D interaction. In particular, 3D prints of corals are used as tangible input devices. The description of the 3D input extension to VTK used in this chapter can be found in Appendix A. Finally, conclusions and possible future work are given in Chapter 7.

1.5 Publications

This thesis is based on a number of peer-reviewed conference and journal publications. Chapters 3, 4 and 5 are based on the following publications:

Quantifying Differences in Skeletonization Algorithms: a Case Study
K.J. Kruszyński, R. van Liere, J. Kaandorp, *Proceedings IASTED International Conference on Visualization, Imaging, & Image Processing (VIIP 2005)*, J.J. Villanueva (ed.), Benidorm, Spain, September 2005, ISBN 0-88989-530-2, 666-673.

An Interactive Visualization System for Quantifying Coral Structures
K.J. Kruszyński, R. van Liere, J.A. Kaandorp, *Proceedings Eurographics / IEEE VGTC Symposium on Visualization (EuroVis 2006)*, B. Sousa Santos, T. Ertl, K. Joy (eds.), Lisbon, Portugal, May 2006, ISBN 3-905673-31-2, 283-290.

A computational method for quantifying morphological variation in scleractinian corals
K.J. Kruszyński, J.A. Kaandorp, R. van Liere, *Coral Reefs*, 26(4):831–840, 2007.

Coral morphometrics: how do simulated corals fit into the Madracis genus?
M.V. Filatov, J.A. Kaandorp, M. Postma, R. van Liere, K.J. Kruszyński, M.J.A. Vermeij, G.J. Streekstra, R.P.M. Bak, *Proc Roy Soc B*, in submission.

Chapter 6 and Appendix A are based on the following publications:

Tangible Controllers for 3D Widgets
K.J. Kruszyński, R. van Liere, *Proceedings IEEE Symposium on 3D User Interfaces (3DUI 2008)*, S. Coquillart, W. Stürzlinger, K. Kiyokawa (eds.), Reno, NV, USA, March 2008, ISBN 978-1-4244-2047-6, 149-150.

Tangible Interaction for 3D Widget Manipulation in Virtual Environments
K.J. Kruszyński, R. van Liere, *Proceedings Eurographics Symposium on Virtual Environments (EGVE 2008)*, Eindhoven, The Netherlands, May 2008.

Tangible Props for Scientific Visualization: Concept, Requirements, Application
K.J. Kruszyński, R. van Liere, *Virtual Reality*, 13(4): 235–244, 2009.

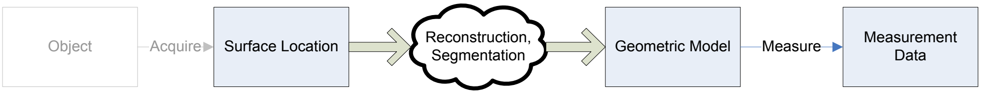
Chapter 2 Related Work/Survey

It is impossible to list here every measuring environment ever created. Instead, a number of different shape measurement environments are discussed, and their relationship to our work is explained using a number of criteria. These criteria are the following:

- How is the object digitized? Different methods of digitizing objects exist, and the resulting data is of different types.
- What processing is involved before the measurements are taken? The acquired data can be processed and transformed into a different representation, and the measured features need to be located.
- What does the system measure, and how are these measurements performed?
- How are the results visualized?
- What role does interaction play in the system? Interaction can be used to measure, view or explore the data, but the data can also be edited during processing.

2.1 Systems

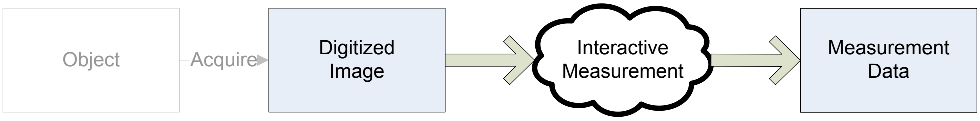
1) In [81], the morphology of articular cartilage is quantified from 3D laser range scanner data by performing measurements on segmented reconstructed surfaces. The measurements include the volume, area and thickness of the cartilage. The method is proposed for establishing ground truth data for validating methods of determining the morphology of cartilage from MRI scans in clinical settings; the thickness of cartilage is close to the resolution of MRI scanners, thus even small inaccuracies in assessment methods significantly affect the results.



The object is acquired using laser range scanning, which produces a large number of points corresponding to part of the surface of the object, from which a surface is reconstructed. To obtain a three-dimensional model of the cartilage, the same joint is scanned twice: on the first scan the surface of the cartilage is acquired, next the cartilage is dissolved off the bone, and the bone surface is scanned again. The two reconstructed surfaces are then combined into a single cartilage model. Thickness, volume and area measurements are then directly and automatically performed on this geometry. The results are shown only as numerical values. Interactive visualization of the model is used in the processing stage of this system: for the reconstruction and combination of the surfaces interactive selection is used, and an expert manually segments the cartilage from the joint.

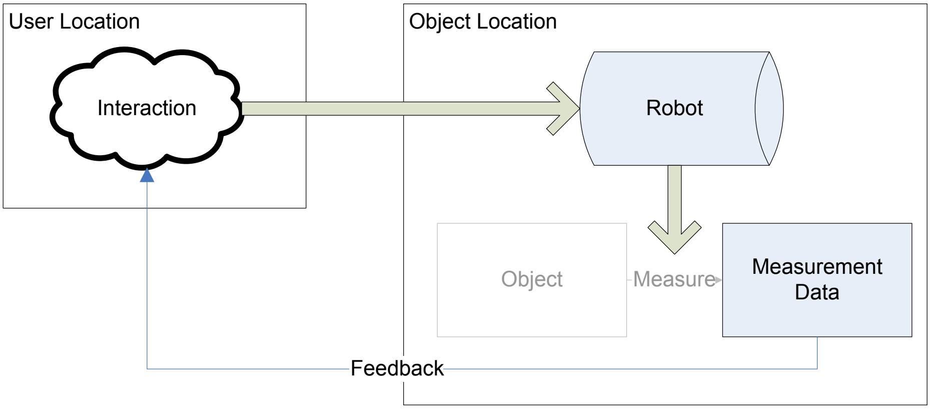
Laser scanning could be used as an alternative to CT scanning of coral colonies. It does however have a major problem: a clear line of view is very difficult to obtain for the inner areas in densely structured colonies, such as the left and middle specimens in Figure 1.3. To obtain a complete 3D model of a whole colony it would be necessary to use a very large number of scanning view directions, which would mean that acquisition and processing would take a very long time. This is assuming that the colony does not have any areas which are completely occluded from all exterior viewpoints, which cannot be expected to be the case for dense colonies.

2) In [89] a number of methods are compared for manually measuring polyp sizes in CT colonography data by measuring the diameter of artificial polyps in a phantom model with known properties. A comparison is made between three different computer workstations and three different visualizations (2D images from axial slices and multiplanar reconstruction, and 3D surface volume rendering) for a number of different polyp sizes and scanner settings. In all cases the measurements were performed directly on 2D and 3D visualizations of the images, using an interactive linear measurement tool provided by the software of the CT workstations. This is an example of how, even when using advanced technology, features are located and measured by hand using interactive tools.



Manual measurements of coral colonies has been one method that was used in the past, for example using photographs. If the number of features to measure is low, manual methods may be sufficient. However, given the sheer number of features to be measured on a single colony, this is quickly becomes a very time-consuming task. Thus, if a CT scan of a colony is already available, it is sensible to apply automatic methods.

3) In [49] a real object is explored remotely. The user is presented with a multimodal haptic VR interface, which shows a 3D model of the object. The user can request various measurement and modeling operations on the object, such as determining the texture of the surface or the acoustic response to tapping the object. A remote controlled robot equipped with several different probes explores the actual object, and the measured data is presented in the VR interface through haptic and acoustic channels. The measured property can be

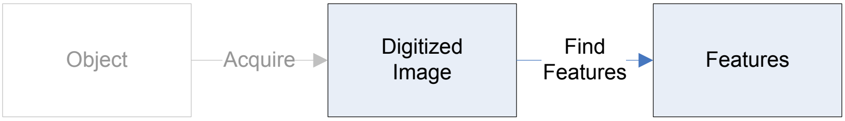


extrapolated onto other areas of the object

This system does not involve actual acquisition of the object; it is assumed that a polygonal mesh model of the explored object is already available, although it is easily conceivable that the robot could produce such a model using for example a laser range scanner or other techniques. The measurements involve various physical properties of the object, and are measured directly on the object by the robot. There is no intermediate representation or feature extraction; however the system does use an advanced haptic and audio interface to convey the measured properties. Also, these properties are not measured in advance and then explored, but they are measured on-demand on the object.

This work is interesting because remote exploration using robotic vehicles is in fact used in marine biology. It would allow a non-destructive analysis of a colony in the original habitat. However, a model of the measured object (coral colony) must first be obtained by some means, which would likely cost much time, thereby negating the advantage of interactive remote exploration.

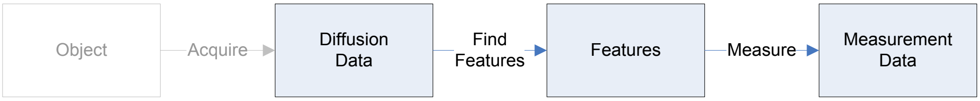
4) In [21] features (lung nodules) are detected automatically in CT scans. The volumetric shape index is calculated for each voxel, and the nodules are then detected by a genetic algorithm performing a template-matching search for spherical objects on the shape images. The relevant features are thus located without first constructing a geometric representation of the data.



The lungs are digitized using a CT scanner. The lung tissue is segmented using an adaptive threshold. This segmentation is only used to limit the area to which further processing is applied. A filter that enhances spherical shapes is applied to the image, and a volumetric shape index is calculated for each voxel. From this shape data regions belonging to spherical objects are determined, and further filtering yields possible lung nodules, i.e., the relevant features. Measuring the located features is not (yet) part of the described system.

Direct shape analysis of features in CT scans is more suitable for distinct isolated homogenous features, such as the lung nodules, than for features which are part of a larger object consisting of heterogenous parts, such as coral.

5) In [4] computational techniques are used to quantify and compare the shape of fiber tracts in brains as extracted from diffusion tensor magnetic resonance images. The three-dimensional tensor image is processed into an intermediate representation (the tracts), which is then quantified. However, the key difference with our work is that the actual tracts in the brain cannot be resolved by the imaging modality; the image data contains only information about the orientation of the bundles, from which a path can be determined. For all intents and purposes, the measured tracts are viewed as curved lines, and all shape descriptors are thus based on orientation and curvature, while the exact spatial extent of the tract is not precisely known.



The acquisition uses diffusion tensor MRI imaging; this technique makes use of the fact that water exhibits anisotropic diffusion in fibrous tissue, with a greater diffusion along the direction of the fibers. From this information the fiber tracts can then be reconstructed. The result of this processing is a large collection of curves. This system applies various metrics to quantify the shape of these curves. It is also proposed to use some of these shape descriptors as a method of selecting particular fiber tracts, and to locate areas of interest on the tracts. The fiber tracts are visualized using 3D visualizations, while the results are either added to this visualization, or presented in various 2D and 3D graphs. Some interaction is used as part of the process of extracting the tracts (e.g., indicating starting and ending regions for tract reconstruction), but the actual measurements do not involve any interaction.

Diffusion tensor imaging technique makes use of a particular physical property of neural fiber tracts in the brain; similar data cannot be obtained from coral colonies. The shapes analyzed here are paths discovered in the object. The analysis is focused on the actual shape of closely related individual curves: similarities and dissimilarities between individual curves; and relative spatial configurations of pairs of curves. There is no analysis of branching patterns. Also, branching objects are more than a set of lines, and possess additional properties (e.g., thickness) which are not addressed in this work.

2.2 Overview

Table 2.1 shows a summary of the systems using the criteria listed at the beginning of this chapter, and compares these systems to our own.

system	**acquisition type**	**processing**	**measuring**	**visualization**	**interaction**
1	surface points	surface reconstruction, model assembly	geometry	3D processing	processing
2	CT volume	none	manual diameter	2D and 3D for measuring	measuring
3	multisensory remote probing	active probing, mapping	haptic and acoustic response	haptic, VR	processing, measuring
4	CT volume	shape enhancement	feature location	none	none
5	DT volume	tract tracing	curve shape	3D tracts, 2D graphs	processing
A	CT volume	object and feature extraction	object shape	2D, 3D, AR, printed props	processing, measuring, analysis

Table 2.1: Features of the different systems discussed in this chapter. The rows labeled 1-5 represent the environments described in this chapter in the order in which they were discussed. The row labeled A represents our system.

Chapter 3 Volume to Skeleton

To measure a coral specimen it must first be digitized. The specimen is acquired as a three-dimensional image. In order to obtain measurements from this image, a number of processing steps are applied to the data. First, noise artifacts in the data are reduced, to mitigate their effect on the rest of the processing steps. Subsequently the coral itself must be located in the volume, by applying a segmentation method. Then the relevant features of the coral must be located in the image, which is accomplished by calculating the morphological skeleton of the image.

Noise filtering and segmentation are explained in the first two sections. In the third section skeletonization is explained. In section four we select an algorithm that is suitable for coral data. This is achieved by an empirical quantitative comparison.

The following steps lead from a grayscale volume to a skeleton:

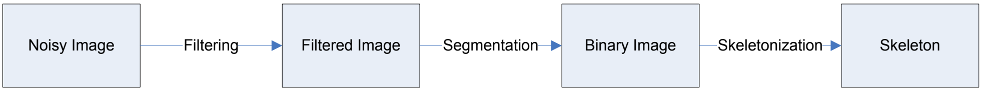


Figure 3.1: Steps from CT scan to skeleton.

1. Noise filtering – noise in the original data is reduced.
2. Segmentation – the object is isolated in the image.
3. Skeletonization – features are extracted.

3.1 Noise Filtering

To isolate objects in images, both two-dimensional and three-dimensional, the images are segmented into several regions using various techniques and criteria such that some of these regions coincide with the object. However, this process is affected by the presence of noise in the image. If the amount of noise is affecting the quality of the segmentation, then a noise-reduction filter can be applied to the image. A number of noise filtering techniques are described in [74].

Noise is defined as the presence of high-frequency components in the image that do not correspond to similar high-frequency features in the original object. It is visible as pixel-sized variation in the intensity of a CT image in areas where the density of the original object or the surrounding air is in fact homogenous. Filtering removes such noise by replacing the value of pixels by some combination of the values of the pixel itself and the pixels surrounding it.

The most obvious source of noise in an image, or volume, is related to the accuracy and sensitivity of the imaging modality. This in turn depends on the particular imaging technology. For example MRI scanners produce much noisier images than CT scanners. The noise is also dependent on various scanning parameters. Scanning thinner slices increases the amount of noise in the resulting image. A lower radiation dose also results in noisier images; for medical purposes the goal is to use the smallest possible radiation dose that still enables a correct diagnosis.

In addition to this basic noise, there is the possibility of additional noise due to interactions between the imaging technology and the particular object being digitized. For example, metal objects distort the magnetic field of an MRI scanner, while the X-rays of a CT scanner are also be scattered to varying degrees depending on the shape and material of the object being scanned.

Coral CT scans in particular contain noise that negatively affects the quality of the segmentation. The actual coral does not solely consist of high-density material, but also of areas of very low density. These areas occur both inside the branches and at the boundary between the coral and the background (the surrounding air). Without noise these areas are easily distinguished from the background, however the noise causes similar density values to occur both in the low-density parts of the coral and in the background. Segmentation of the image then either incorporates parts of the background into the coral, or conversely parts of the coral are missing. This affects the measurements, as the segmented object does not properly correspond to the actual coral. The substandard segmentation further also affects the quality of the subsequent skeletonization of the coral, introducing further degradation of the measurements.

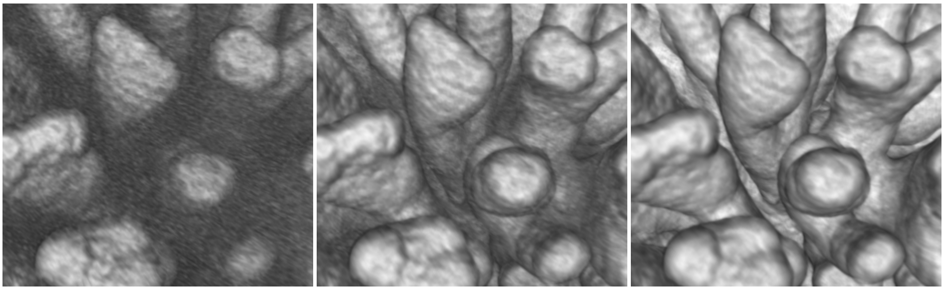


Figure 3.2: Volume rendering of coral with noise. On the left is the original CT scan, the middle and right images show the effect of an increasing amount of noise filtering. The noise is visible as the dark haze between the branches.

The images in Figure 3.2 show the original CT scan and the same data after noise filtering. Most noise can be found at the center of scanned colonies, where the scanner rays must pass through many branches. This is in contrast to for example the base of a colony, where only a single branch is scanned and little noise is present. The noise is very likely due to significant scattering of the X-rays. As mentioned before, there is little experience in scanning coral colonies, and it is possible that better, less noisy images could be obtained if different parameters (such as a higher radiation dose) were to be used for the scan or the CT reconstruction, or if a different scanner were to be used. The structure of coral is quite unlike (human) bones; it is a very porous structure of dense material with many fine details.

To quantify the noise in the image we consider the mean and standard deviation of the background density values in two areas, one at the center of the colony and one at the edge of the scan, on sample slices from each specimen. These areas are compared to the same areas after filtering, corresponding to the left and center image of Figure 3.2.

The standard deviation of the background of an ideal, noise-free image is always 0, as the background is supposedly a uniform area consisting of only minimum values. For CT scans the minimum value is -1000.0. Any deviation from the minimum value in an area of the CT scan containing nothing but air is the result of noise.

The slices and areas are shown in Figure 3.3. In addition, a slice containing a single branch cross-section of the third coral is also measured; this is shown in Figure 3.4. The results are shown in Table 3.1.

What can immediately be seen is that only the edge of the single-branch slice is free of noise; all other sampled locations do contain noise, including the center of the single-branch slice. Also, in all cases there is much more noise present in the center of the slices than near the edge.

It can be seen that there is less noise in slices where less dense matter was traversed by the radiation of the scanner, and thus where less scattering occurred. Object 1 has thinner and much less densely spaced branches than the other two objects, which results in somewhat less noise. Also, the center of the single-branch slice of object 3 contains less

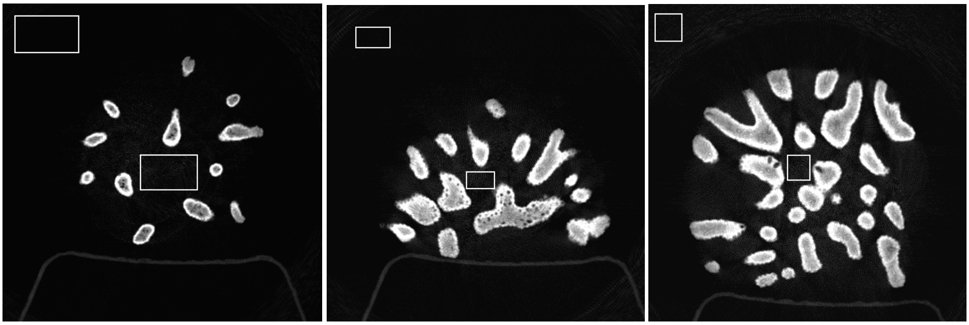


Figure 3.3: Background noise measurement for each of the three objects. The slices are from the center of the objects, in the X-Y plane, from the non noise-filtered data. The red boxes represent the actual areas where the measurements were performed.

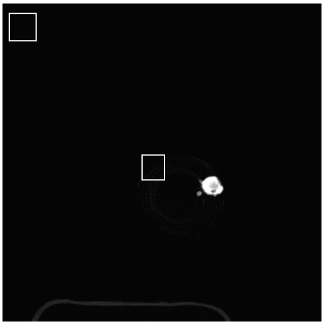


Figure 3.4: Additional background noise measurement. This was performed on a slice with only a single coral branch, from the scan of the third object. The red boxes represent the actual areas where the measurements were performed.

noise than the edge of the multi-branch slice of the same object.

Some filtering methods are less suitable for coral images. The branches of corals can grow very close together, yet many species exhibit some sort of mechanism that prevents the fusion of branches. Given current scanner resolutions, it is not uncommon for branches to be a voxel apart. When noise filtering is applied to CT scans of coral that has such closely spaced branches, these branches may appear fused in the resulting images. The resulting topology of the coral can thus be altered by the filtering.

Given that the thickness of the coral branches is measured, it is also important that the edges of the branches remain as close to the original location as possible. Indiscriminate filtering has a tendency to move such edges, thus affecting the results of any subsequent measurements performed on the images.

	Area	**Before filtering**		**After filtering**	
		mean	***std***	***mean***	***std***
Object 1	center	-932.39	104.93	-950.08	49.45
	edge	-997.28	9.90	-998.27	4.72
Object 2	center	-882.86	161.09	-910.20	77.01
	edge	-991.99	14.51	-994.42	7.53
Object 3	center	-877.39	148.33	-896.43	83.67
	edge	-964.93	56.34	-974.75	21.75
Object 3 ***single branch***	center	-986.80	33.63	-992.05	17.82
	edge	-1000.0	0.0	-1000.0	0.0

Table 3.1: Noise measurements before and after filtering. On each slice an area near the center and an area near the edge were measured.

There are many different algorithms available to reduce the amount of noise in an image, by altering the value of a pixel through a combination of the value of neighboring pixels. Some of these filters significantly alter the contents of the image; especially object borders can be affected. Others are designed to leave sharply defined edges intact and focus only on smaller noise artifacts.

3.1.1 Filtering Methods

- **Mean and Gaussian filtering**

The simplest method to reduce noise is an averaging filter. The value of each filtered output pixel $f(x)$ is determined by the mean

$$f(x) = \frac{1}{N}\sum A(x)$$

where $A(x)$ is an area centered on the corresponding input pixel. The resulting output image contains much less noise than the original, but in effect much of the resolution of the original image is lost.

Gaussian filtering uses a weighted average

$$f(x) = \frac{1}{\sum_{i \in A(x)} G_\sigma(i)} \sum_{i \in A(x)} i \cdot G_\sigma(i)$$

where $G_\sigma(x)$ is a Gaussian function with mean 0 and standard deviation σ. This leaves more noise in the image than mean filtering; however the loss of image sharpness is much reduced.

Both these methods make sharply defined edges in the image much softer, which makes subsequent segmentation of the image more difficult, as this relies on clear sharp boundaries. In addition, closely spaced parts of an object may become fused in the resulting images. This is particularly problematic for coral data, which is characterized by tightly spaced branches. Fusion of the branches changes the topology of the object and has to be corrected afterwards.

- **Anisotropic diffusion**

Anisotropic diffusion [63] considers the original image as the initial condition for a heat diffusion calculation, with the value of each pixel viewed as a temperature. The diffusion of this heat is then simulated over a number of time steps, thus the values of the pixels change in the same way in which the heat would spread. A standard Gaussian filtered image is in this context the result of a certain amount of heat diffusion with the diffusion occurring uniformly throughout the image.

The main feature of these algorithms is that the speed of the diffusion of the heat across the image, and thus the amount of filtering, is not uniform but varies according to a criterion that differs between algorithms. This allows for noise filtering that can preserve or even enhance edges in the image.

The various algorithms of this type use varying criteria for the diffusion speed and different methods of calculating the diffusion. They are all significantly slower than normal filtering methods, and also do not lend themselves very well for parallel execution.

One criterion that can be used for diffusion is the gradient of the image. With increasing gradient there is less filtering, which preserves the sharp edges while smoothing out the noise, which typically has only a low gradient. Another criterion takes into account the curvature of edges in the image. The higher the curvature of an edge is, the more likely it is that this is in fact noise rather than an edge, thus more filtering is applied.

Edge preservation and edge enhancement are desirable characteristics when preparing noisy images for segmentation, which makes anisotropic diffusion filters very suitable for de-noising CT scans of coral. We have therefore opted to use anisotropic diffusion to filter the noise from the scans.

3.2 Segmentation

Segmentation is the process of partitioning an image into regions of which some, hopefully, coincide with the object of interest. An overview of segmentation techniques can be found in [24]. A large number of different methods exist for this task, ranging from fully manual to fully automatic. The difficulties with segmentation arise mainly from how sharp the edges of the object are, and how homogenous the pixel values of the object are.

It is difficult to obtain a good segmentation of coral data. The material is porous, and the density of the material varies. The coral is not limited to the high density parts of the image, while the low density parts of the coral do sometimes not differ significantly from the background. In part this is due to various artifacts in the coral, which strictly speaking are not part of the coral, but which for subsequent analysis purposes must be considered as such. One example of this is that branches of the coral might have been broken and reattached using glue. This can be seen in the right image of Figure 3.6. The break is clearly visible, and has a density similar to the background; for analysis, it should be considered to be part of the coral. However, a similar low-density gap between high-density areas in the image might very well be simply due to the proximity of closely spaced branches, in which case the gap should not be part of the segmented coral.

Figure 3.5 shows the segmentations of our three specimens.

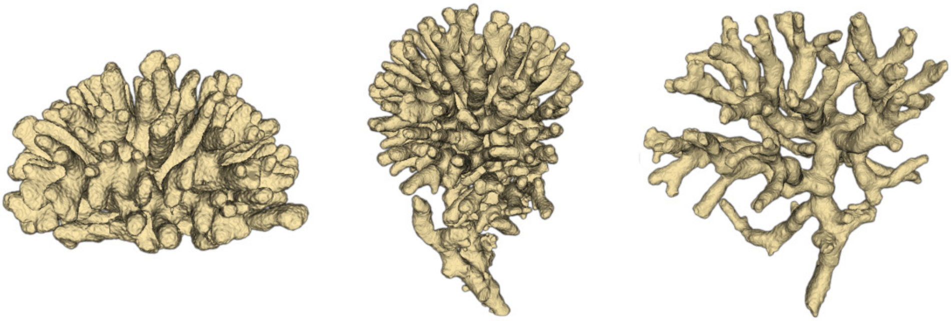


Figure 3.5: Segmentation of the three specimens. Shown is the iso-surface of the segmented region.

3.2.1 Segmentation Methods

There is no gold standard segmentation method. In some cases, the gold standard is considered a manual segmentation performed on the data by an expert. The volume is viewed slice-by-slice and each voxel that the expert considers to belong to the object in question is manually marked as such, either pixel-by-pixel or by drawing and filling contours. Needless to say, this is a very laborious task, the results are somewhat subjective and are likely to differ between experts, and perhaps even for the same expert and data, if the segmentation were to be performed twice.

A large variety of automatic methods exists. Here we give an outline of a few commonly used approaches.

- **Thresholding**

The simplest method of automated image segmentation uses one or more threshold values to divide the image into regions. Thus all pixels or voxels above or below the threshold value, or between two such values, are considered to belong to the object of interest. One method for the automatic selection of a threshold value is described in [60].

Thresholding is a rather inaccurate method that only works well for images with high contrast between the object of interest and other items in the image, and the results are highly affected by noise. However, it is extremely fast to calculate, and it is thus commonly used to extract features from video images in real-time. In addition, the method is also used to locate suitable starting regions for other more complex segmentation methods.

Unfortunately the noise sensitivity of this method makes it less suitable for use on the coral CT scans.

- **Watershed segmentation**

The watershed transform views an image as a topological height map. This map is then flooded starting at local minima, while keeping track of where each flood originated, thus preventing the different flooded areas from merging. This partitions the image into two areas: the disjoint catchment basins and the watershed lines between the basins. Many varieties of this scheme exist. A more elaborate explanation of the method, and an algorithm, can be found in [83].

This transform is used to segment an image by applying it to the gradient of the image instead of the image itself. This segments the image into areas with similar values, while the watershed lines will be located where the gradient is highest. In addition the technique is often used with markers in the image to indicate where flooding should start, improving the segmentation. Such markers can be placed manually, or be determined automatically using a suitable heuristic. There will be as many regions as there were markers.

The technique does not work well with noisy images such as our coral scans, as the resulting segmentation is highly fragmented into small regions. This is mitigated by using a hierarchical approach. From the initial segmented image a hierarchy is established among the regions based on the relative height of the watershed walls between the regions. This new image is then again processed using the watershed transform. The result is that regions with watershed walls with similar height are merged, and a segmentation with higher level features is obtained.

The method is easily extended to 3D images and beyond, even though the concepts of a flood of water and terrain height are then no longer appropriate. It is then best compared to expanding gas in a porous medium.

- **Region growing**

Another collection of methods is based on growing regions from seed points. One such method is described in [2]. Starting from either the seed points, or a region around the seed points, neighboring pixels are added to the region. The main difference between algorithms of this kind is in the different criteria that decide whether a pixel is to be added to a region. These can be fixed criteria, such as whether the value of the pixels falls within a certain pre-determined threshold, or it can be based on statistical analysis of the values of the pixels that are already contained within the initial region. The process stops when there are no more neighboring pixels that match the criterion.

If the criterion is based on the contents of the region, it may be necessary to use several iterations to reach a good result. Often selecting more strict criteria and repeating the process several times will yield better results than using less strict criteria initially, which may lead to the extraction a region that includes areas that are not part of the object.

The advantage of using region growing algorithms for segmenting coral is that it yields connected and mostly contiguous regions, while ignoring unconnected parts in the image where the density values may be similar to those of the coral. It also allows for an interactive process where the user can select locations on the image that are not yet segmented, and the resulting segmented regions are added to the total segmentation; this method is very similar to how fuzzy selection tools function in many image manipulation programs. For these reasons we choose this method to segment the coral data.

3.2.2 Post-segmentation Processing

As mentioned, automated segmentation of the data may lead to less than satisfactory results. This can be the result of unwanted artifacts in the original data, or even in the object itself. If such artifacts cause problems with the extraction of the skeleton, they should be removed from the data.

The simplest case is where artifacts can be automatically located and fully isolated in the data; it is then possible to repair the data without user intervention. However, this is seldom the case, and two more commonly occurring scenarios are artifacts that can be detected but not isolated, or that cannot be detected reliably, and artifacts that cannot be automatically detected at all. In both cases interactive data exploration and editing tools should be provided to the user. If the presence of artifacts can be detected, but the artifacts cannot be reliably pinpointed in the data, then the system can indicate the artifacts and propose possible repairs to the user, who can then adjust the repairs as necessary, or reject them if the detected artifact is actually a false positive.

Here we discuss an artifact that we encountered in our coral data, which required additional processing after segmentation. The artifacts could not be completely reliably detected, but a detected artifact could be correctly repaired. Thus the necessary interaction was limited to confirming whether each artifact had been properly classified as such.

- **Holes**

A curious feature of at least the *M. mirabilis* coral species is that after segmentation some of the branches are hollow. In addition, there are open connections between this hollow inside and the non-coral parts of the image. This makes these holes difficult to locate and correct.

- **Origin**

The core of the coral branches is rather porous compared to the outer layers. This is a result of how the coral grows. A polyp may continue to grow away from its original location. Its length does not increase, but it continues to secrete new corallite while the old corallite remains how it was. Thus in essence most polyps live on top of the long tubular corallites that may extend all the way down to the base of the colony.

The resolution of the scan is not sufficient to distinguish the calcified material of the corallite and the empty space in between, inside the branches; instead the average density is digitized. This results in low density values.

Due to various reasons discussed before there is a noticeable amount of noise in the center of the coral colony, and a higher average density value for the air in the center between the branches. In fact, this value is approximately the same as the value for the low-density corallite core of the branches. This might cause cavities in the resulting segmentation.

In addition there are various low-density connections between the low-density core and the surrounding background. Some of these connections are actual holes in the coral, resulting from predation or taking of samples for analysis, while others might be particularly large, porous or damaged corallites.

For these reasons the segmentation algorithm cannot distinguish between the background and the inside of the coral using either density values or region connectedness.

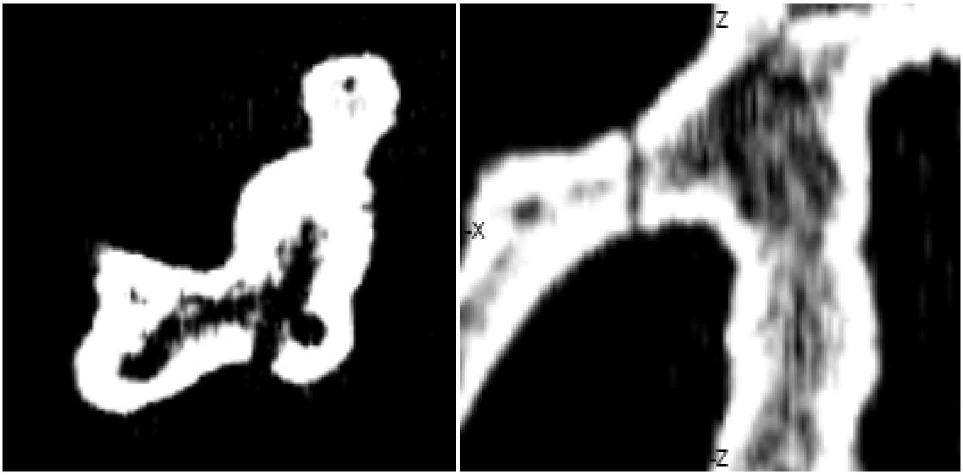


Figure 3.6: Difficulties in coral images. Both cross-sections of coral show the low density core. The left image shows a hole connecting this low density area with the background. The right image shows where a branch has been fractured.

- **Filling**

While it is known how to fill a cavity that is completely enclosed inside the object, filling holes connected to the outside presents quite a challenge. Most work in the area of hole-filling is focused on either patching clearly defined gaps in mesh surfaces, or retouching gaps in 2D color images. In [20] holes in a polygonal mesh are closed by extrapolating the existing surrounding surface. However, gaps in polygonal meshes are easy to locate, as these consist of paths along polygon edges which are used by a single polygon only. In [6] areas in color images are automatically filled using surrounding parts of the image, however, the areas must be indicated manually.

A cavity in an object is defined as a set of connected background voxels that is surrounded by object voxels. These background voxels thus form a ‘bubble’ inside the object. The common approach to fill such cavities is by using a flood-fill algorithm. The background of the image is flooded with a new value, starting at a non-object voxel, for example a corner of the image. After the image is flood-filled in this manner, the only remaining areas containing the background value are the cavities in the object, thus their location and extent is now precisely known. Thus, if it is known that an object is not supposed to contain any cavities, these can be found and filled.

Holes are cavities that have a connection to the ‘regular’ background in the image. Due to this connection, it is not possible to locate them using a flood-fill algorithm. As there is no boundary between the hole and the background, the hole will be filled along with the background.

However, if the volume is divided into 2D slices, it can be seen that in many of these slices the part corresponding to the hole is actually an island of background values fully enclosed by foreground values, as would be the case with a regular cavity. By combining this information from many slices, cut in different directions from the volume, it is possible to locate most of the voxels belonging to the hole.

In our approach, first all cavities are filled with a regular 3D flood-fill. Then the remaining holes are filled. This is accomplished by a slice-by-slice 2D flood fill, with slices perpendicular to the primary axes, as well as slices perpendicular to the side diagonals of the volume, thus 9 slicing directions in total. The background in each slice is flood-filled with the foreground value, and then inverted, so the only remaining foreground voxels in the slice are those that belong to a hole. These slices are then recombined into a new volume using a logical OR operation. The whole process is performed a second time on the output of the first run, to fill some remaining gaps that could not be filled in the first iteration.

Since it is possible that this method would fill up areas that would not be considered holes, the fillings must be checked manually by the user. This is done by showing the user each filling, along with the original object. The user can then decide for each filling whether it is correct or not, and only the correct ones are then combined with the object. Because most of the fillings created by this method are very small, and not likely to be very significant, they are presented in order of decreasing size. The user can then decide that any filling smaller than the current one will be insignificant, even if incorrect, and stop checking the remaining fillings.

3.3 Skeletonization

A skeleton is a transformation of an image whereby the objects in the image are reduced to a collection of lines going through their centers, originally proposed in [9]. This reduced dimensionality makes it much simpler to locate and analyze features in an image.

However, there is no single precise mathematical definition of this transformation, so that each algorithm produces different results. A survey of skeletonization algorithms can be found in [39]. Different classes of skeletonization algorithms have different general definitions of what it is that they attempt to compute, and even when two algorithms use the same definition of the skeleton they may arrive at different results.

The following features are important characteristics of skeletonization algorithms:

- The skeleton should be located at the center of the object;
- The skeleton should be thin (i.e., one pixel or voxel thick);
- The results should be invariant under various transformations, such as rotation, translation and scaling;
- The topology of the object should be preserved;
- The result should not change in the presence of noise.

The skeleton of an image is only an alternative representation that is used for various purposes, and not a goal in itself. Thus when comparing algorithms, it is sensible to do this by judging their suitability for the intended purpose. This can mean that a perceived advantage of an algorithm turns out to be irrelevant.

Since a computed skeleton in many cases is still a binary image, it is helpful to convert the skeleton into a graph in which the pixels or voxels that belong to the skeleton are the vertices, and the connections between them are the edges. The method used to perform this conversion is outlined in section 4.1.

3.3.1 Skeletonization Methods

The approaches to computing skeletons can be broadly classified into four categories; topological thinning, distance transform algorithms, wave propagation based algorithms, and Voronoi diagram based algorithms.

- **Thinning**

Thinning is a method of obtaining the skeleton of an object by iteratively removing each successive outer layer of an object's voxels, until the object is just one voxel thick. An overview of the concept can be found in [47]. The input of an iteration n is the image I_n, and the output is the image I_{n+1}. Each iteration of a thinning algorithm consists of one or more sub-iterations. In each sub-iteration the algorithm selects a set of candidate voxels with the foreground value 1, and determines which voxels are removed (i.e., set to the background value 0). The algorithm stops when $I_{n+1} = I_n$, i.e., when none of the candidate voxels could be changed. The algorithm considers all of the remaining object voxels to be part of the skeleton.

In a parallel sub-iteration the algorithm considers each candidate voxel separately, independently from the others, and then changes all suitable voxels at once. In a sequential sub-iteration on the other hand the algorithm considers and removes the voxels one at a time, thus the removal of each voxel is influenced by voxels removed previously in the same sub-iteration.

The *neighborhood* of a voxel is an important notion in thinning algorithms. If the voxels of an image are considered to be adjacent cubes, centered at the voxel locations, then the 6-neighborhood of a voxel consists of all surrounding cubes with which the voxel shares a face; similarly, the 18-neighborhood consists of all face and edge neighbors, and the 26-neighborhood consists of all face, edge, and vertex neighbors. Two voxels are called n-adjacent if they are in each others' n-neighborhood.

- **Distance transform**

The distance transform of a (binary) image gives for each voxel the distance to the image background, as outlined in [10]. Distance transform based skeletonization methods search this information for local maxima; such voxels are part of the skeleton. One such method is given in [56]. Although skeleton voxels found by these methods are always exactly in the center of the object, the detection of these voxels is non-trivial, and they are usually not connected, requiring additional methods to obtain a connected skeleton.

The skeletons of distance transform based algorithms are usually more suitable for reconstruction of the original object rather than shape analysis, and are very sensitive to noise. The skeleton is likely to continue to the edge of an object, and especially in objects with flat sides or small perturbations there are many branches present that enable a full reconstruction of the original object, but which are not significant for broader analysis of the shape. An added difficulty is that there is no guarantee that the topology of the object will be preserved by the algorithm.

- **Wave propagation**

Wavefront propagation methods calculate the arrival time of a wavefront propagating from a starting point in the object to the outer reaches, and trace back the path of this front to create a skeleton [16]. They can be made highly insensitive to noise, and they can produce smooth continuous skeletons, but they will not always produce a skeleton that is centered inside the object, as the method has a tendency to cut corners.

The starting locations of the traceback, and thus the endpoints of the skeleton, are either selected manually, or consist of the voxels where the wavefront ceased propagating, with additional methods to limit their numbers.

Normally the resulting skeleton would be a set of unconnected paths leading to the origin of the wavefront. To obtain a connected branching skeleton the paths can be merged when they come close together, however the parameter governing the merging distance has a very large influence on the exact location of the junctions. This is a major problem if the junctions in the skeletons are used for measurements, as is the case with coral.

- **Voronoi diagrams**

Voronoi diagrams are created by choosing a set of voxels and partitioning the image into regions closer to one of these voxels than to any of the other voxels. When these voxels are chosen on the boundary of the object, the skeleton can be constructed from the boundaries dividing the regions, as outlined in [54]. While Voronoi diagram based methods can produce good skeletons, the step from diagram to skeleton is mostly based on heuristics, and can be computationally complex, which makes the method less suitable for large, complex objects. Also, the algorithms are more commonly applied to polygonal data rather than image data.

3.4 Choosing algorithms experimentally

Given the differences between algorithms, choosing a suitable algorithm requires an objective comparison between the algorithms.

We consider how to choose a skeletonization algorithm, as the output of these algorithms is directly used for measuring, and the output of these algorithms varies significantly. The methodology applies equally to the algorithms used in the preceding steps, but the different algorithms, when correctly applied, exhibit less differences in the output and thus have a much smaller effect on the measurement results.

An objective comparison is needed when evaluating the suitability of algorithms for a specific application. This can be achieved through quantitative comparison of algorithm output using application-relevant metrics.

One or more quantifiable metrics of the skeleton must be defined. These metrics should be related to the application for which an algorithm is to be selected. This consists of the metrics directly used by the application, but it can also include metrics such as the number falsely identified or overlooked features.

A test dataset for which the correct metrics are already known is then needed. This can be an existing dataset that has been previously measured, or the data can be generated. Here it is important to not use a skeleton resulting from some skeletonization algorithm to directly obtain the reference metrics, as this will invalidate the comparison and only give a comparison with the original algorithm.

The candidate algorithms are then applied to the data, and the resulting skeletons are measured using the previously defined metrics. These results can then be compared, and a choice can be made.

We have compared three different algorithms in a case study with a synthetic object. Two of these are thinning algorithms, one based on template matching and one based on component counting. The third algorithm involves wavefront propagation. This has been followed up by a more elaborate comparison of one of these thinning algorithms and the wave propagation algorithm using test data generated from real-world objects.

The following algorithms were compared:

- **Palágyi**

Palágyi and Kuba describe a border sequential thinning algorithm in [61]. It repeatedly compares border voxels and their neighborhood with a set of templates, and removes them if there is a match.

Each iteration of this algorithm consists of six parallel sub-iterations. Each sub-iteration only considers object voxels, which have a neighboring background voxel in one particular direction in the 6-neighborhood of the candidate voxel. The templates are appropriately rotated or mirrored for each direction. Each voxel matching one of the templates is removed. The order of the directions is chosen so that the object is thinned evenly on all sides.

- **Xie**

In the algorithm of Xie, Thompson and Perucchio, described in [87], each iteration consists of six directional parallel sub-iterations, followed by a repeating sequential sub-iteration. As in Palágyi's algorithm, the directions are chosen to ensure the object is thinned evenly.

The parallel sub-iterations remove candidate border voxels by classifying neighboring voxels, and counting the number of connected components formed by the voxels of each class. If the numbers of components satisfy certain requirements, the voxel can be removed.

The sequential sub-iteration attempts to remove voxels that were considered but not removed in the parallel sub-iterations. It uses a slightly different set of conditions, and checks the voxels in a specific order, to avoid removing too many voxels.

This algorithm also includes a pre-processing stage for reducing noise in the input image. Single-voxel protrusions are removed, and single-voxel indentations are filled.

- **Deschamps**

The algorithm of Deschamps, provided in [22], uses a modified Fast Marching Method (explained in [72]) to propagate a wavefront from a starting point s throughout the object.

The Fast Marching Method calculates the front arrival time $T(x)$ for all voxels. This wavefront is then traced back to s using the Gradient Descent method, which directly results in the skeleton graph G.

The speed of the wavefront at each voxel is determined by the speed function $V(x)$. For this purpose we used $V(x) = \left(D(x)\right)^2$ where $D(x)$ is the distance transform [10] of the data set; this ensures that the skeleton is centered within the object, as this causes the wavefront to propagate much faster in the center of the object. A more uniform propagation speed causes wavefronts to take the shortest path and cut corners, resulting in off-center skeletons.

The points $t_0 \dots t_n$ from which the front should be traced back to s are determined with the aid of a modification to the Fast Marching Method. In addition to $T(x)$, the algorithm calculates the length $d(x)$ of the path $x \dots s$ that the wavefront followed to arrive at x. The object is then divided into connected regions R_i in which $na \le d\left(x_{R_i}\right) < (n+1)a$ for some interval size a. A spanning tree is determined for the connections between R_i, starting at the region containing s; from each leaf node region a traceback point t_n is selected by $\operatorname{argmax}_{t_n} d(t_n)$.

The backtracking does not follow the voxels; instead the object voxels are considered to be grid points where $T(x)$ and $\nabla T(x)$ is known. At each step, $\nabla T(x)$ is interpolated for the current location.

Because the backtracing would produce a set of unconnected lines, each new trace is halted and connected to another trace if the distance is smaller than a specified threshold value.

3.4.1 Measurements

Several measurements were performed on the skeletons. Many of these measurements are commonly used by coral biologists. They are described in more detail in section 4.2.1.

1. Minimum thickness (*da*, *a*-spheres).
2. Maximum thickness (*db*, *b*-spheres).
3. Branching angles (*b_angle*).
4. Branching rate (*rb*).
5. Branch spacing (*br_spacing*).
6. In addition to these measurements used by biologists, the ratio of the length of the skeleton graph between every two successive branching points, and the Euclidean distance between these two points, was calculated. For a segment of the skeleton consisting of the points $p_0 \dots p_n$, with branching points p_0 and p_n, the formula is:

$$\frac{\sum_{i=0}^{n-1} \|p_{i+1} - p_i\|}{\|p_n - p_0\|}$$

The original skeleton consists of straight lines, with ratio 1.0, therefore this ratio indicates how straight the calculated skeleton is.

7. The Horton-Strahler ordering [33,76] is a topological ordering of the branches of a tree. It is described in section 4.1.1. As metrics we take the number of the highest order, the number of branches of order 1, and the average length of branches of order 1.
8. The average angle between successive bifurcation planes. To find this angle, first two successive branching points must be located, each of which must have two b-spheres. At each branching point a, a plane p is constructed passing though a with normal $n = \overrightarrow{ab_1} \times \overrightarrow{ab_2}$ where b_1, b_2 are the centers of the b-spheres. The angle between the two planes is then calculated. The metric is the average of these angles.
9. The total number of branches in the graph: $|B(G)|$. This metric does not require the skeleton to be a tree.
10. Torsion and curvature. The torsion and curvature of the graph is determined at each vertex of degree 2. The other vertices are not considered as it is not possible to determine the torsion and curvature at such points. To determine the torsion and curvature, a Frenet frame [8] is constructed from the vectors corresponding to the two edges at the vertex. The roots of the mean squares of the torsion and the curvature are then used as metrics.

3.4.2 Synthetic Object

We have conducted a case study comparison of the three previously described skeletonization algorithms with a single fully synthetic object with various noise levels as the ground truth. The comparison used measurements 1 through 6.

The data used for this study is a set of 3D binary images of a fractal-like tree, to which increasing amounts of noise have been added. A synthetic object was chosen in order to have a reference skeleton, or ‘Ground Truth’, to which the computed skeletons could be compared.

To create the data sets, first a polygonal line model of a tree was generated. The model starts with a vertical line. From the top of this line, two new lines continue, one at 55° to the left, the other at 55° to the right from the initial line, in the XY plane. The length of the new lines is 0.8 times the length of the initial line. This process of adding new, shorter lines is repeated seven more times, with each two new lines constructed in a different plane, rotated 65° to the right around the parent line, relative to the previous plane.

Next, around these lines a set of round polygonal tubes was constructed, with a linearly decreasing diameter, such that the diameter at the base of the tree was 10 times the diameter at the end; the diameter at the base was set to 0.2 times the length of the first segment. Spheres were added at the root and the endpoints of the tree, with the same diameter as the tubes, to create round endpoints. This object is shown in Figure 3.7.

The polygonal model was then voxelized as a binary volume, with the inside filled with the value 1 while the background was set to the value 0.

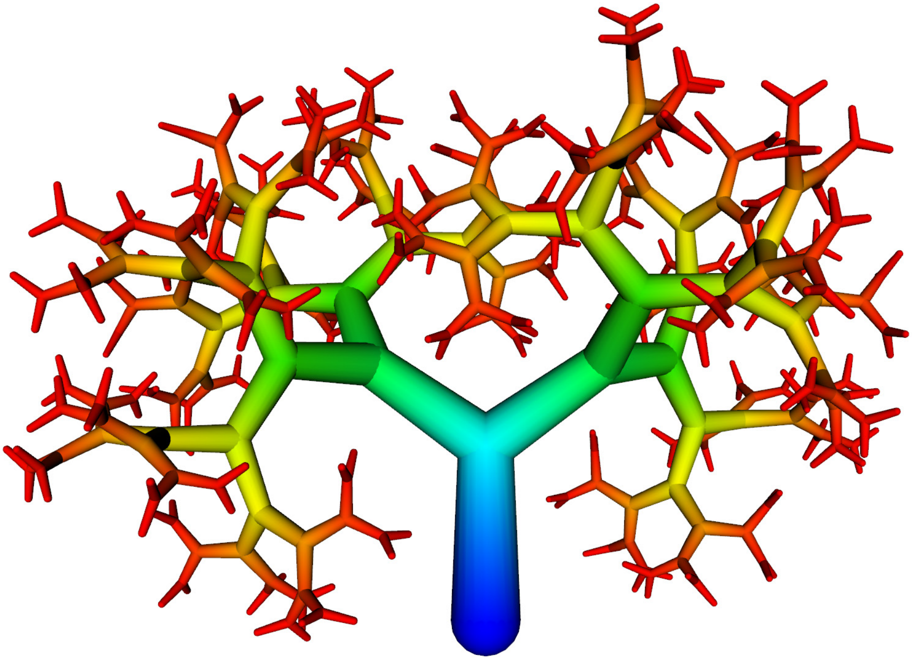


Figure 3.7: The polygonal model of the test object. Color indicates the diameter of the object.

Additional data sets were produced by adding increasing amounts of noise to the initial data, around the boundary between object and background. To add the noise, first a volume of the same size as the data set was filled with random floating point values between 0.0 and 100.0. To generate a particular noise level *l*, first all points from this volume with a value less than *l* were selected. All corresponding points in the binary image, which were found to be 6-adjacent to at least one voxel with value 1, but themselves had value 0, were then set to 1, producing an intermediate image.

Next, from the random data all points with a value greater than 100.0-*l* were selected. Each corresponding 1-valued voxel from the intermediate image, which was 6-adjacent to at least one 0-valued voxel, was then set to 0 in a final image. For our analysis we have chosen noise levels 0, 1, 2, 4 and 8.

This method adds noise to the surface of an object, thus increasing the roughness of the surface. It is assumed that segmentation of a CT-scan of a coral would result in a similar rough surface, if there is noise present in the scan.

3.4.3 Synthetic Object Results

The synthetic object was voxelized to create a 3D binary image with a resolution of 360×300×240 voxels. The algorithms of Palágyi, Xie, and Deschamps were then used to obtain skeletons of the images.

While the algorithms of Palágyi and Xie are turn-key solutions, the wavefront propagation algorithm requires many parameters. It was used with the following settings: points were frozen when the distance to the starting point was less than 0.5 times the maximum distance; the distance interval for endpoint detection was 6.0; the minimum region size was 5 voxels; the gradient descent step size was 0.5; and the minimum distance between traces was 3.5.

The graphs in Figure 3.8 show the average values of each measurement, for each algorithm at each noise setting. In addition, the results of the same measurements for the original skeleton, which was used to construct the test object, are shown in each graph. These measurements were made by subdividing the lines of the original skeleton until each of these smaller lines had a length of less than 0.5, and using these lines as a skeleton graph for measurements.

The graph of the average radius of the a-spheres shows that the Xie results are the best for this particular measurement, but the difference with wave propagation is minimal. Palágyi shows erratic results, and only performs well for the highest noise level.

For the average radius of the b-spheres, Xie performs slightly worse than the other algorithms at lower noise levels, but at highest noise level Palágyi produces a quite different average, while wave propagation performs the same as Xie.

The angles between branches are fairly constant across noise levels for Xie and wave propagation. The angles with Xie are a little too high, while they are a bit more too low with wave propagation. Palágyi gives higher angles than Xie at lower noise levels, but gradually decreases to almost the same angle as wave propagation at the highest noise level.

For the branching rates, both Xie and wave propagation give results very close to the reference skeleton. The results with Palágyi are a little worse without noise, but deteriorate rapidly when noise is added.

The branch spacing is consistently too high with wave propagation, and much too low with Xie. Palágyi with little noise produces the best results, but is still too low. With more noise, the average value becomes much too low.

The length/distance ratio between branching points is reasonable consistent between noise levels. The thinning algorithms have a much higher ratio than the wave propagation algorithm. This is mostly due to the voxel-based nature of these algorithms, which increases the length of a diagonal path between more distant voxels due to aliasing, as opposed to the smooth lines produced by the wave propagation algorithm.

Table 3.2 shows the number of a-spheres found by each algorithm for each noise level. Recall that an a-sphere indicates a junction point in the skeleton. The table shows that the Palágyi skeleton is very sensitive to noise, while the Xie algorithm is less sensitive. The number of a-spheres found by Deschamps algorithm remains the same as that of the reference skeleton.

	level 0	**level 1**	**level 2**	**level 4**	**level 8**
Reference	255	255	255	255	255
Xie	255	255	255	257	259
Palagyi	267	275	287	341	517
Deschamps	255	255	255	255	255

Table 3.2: number of a-spheres for each noise level.

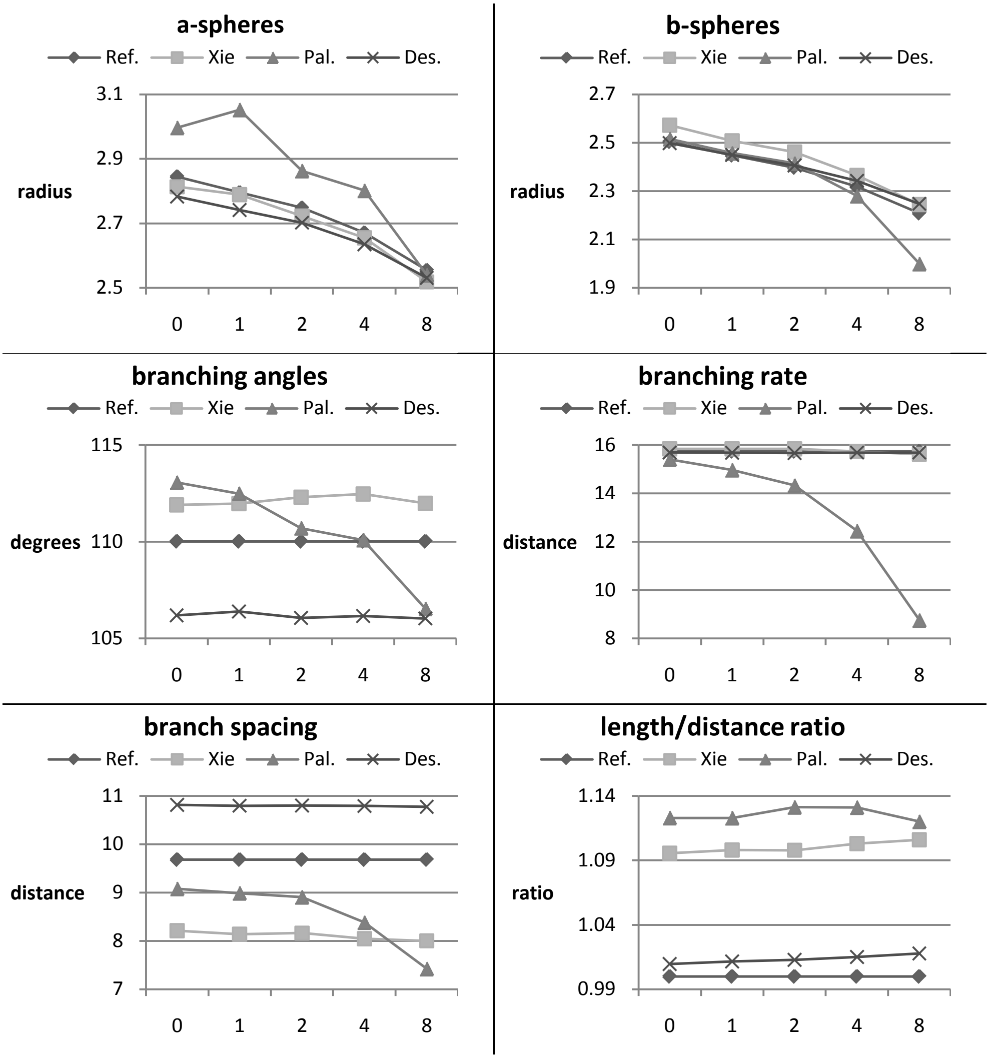


Figure 3.8: Graphs of each measurement. Shown are the reference skeleton (Ref.), Palágyi skeleton (Pal.), Xie skeleton (Xie) and Deschamps skeleton (Des.). The x-axis denotes noise settings 0, 1, 2, 4, 8. The y-axis denotes the measured values.

3.4.4 Real World Objects

In a subsequent study, a large number of different semi-synthetic objects have been generated by perturbing centerline skeletons of a number of existing objects and creating new objects around these skeletons, with various levels of noise. These objects were then subjected to different skeletonization algorithms and the resulting skeletons were measured.

The algorithm of Palágyi was not considered in this study, as the previous study indicated it was too sensitive to noise to be of practical use for analyzing coral data.

Measurements 1-5 and 7-10 were used. Measurement 6 was not used, as wave propagation algorithms will inherently produce smoother skeletons than thinning algorithms.

Three different input data sets were used:

- A CT scan of a specimen of the Madracis mirabilis stony coral. This data set had a resolution of 512 x 512 x 512 voxels;
- An MRI angiography data set of a human head. The resolution of this data set was 416 x 512 x 112 voxels;
- A volumetric image of the roots of a pine tree; this data set is distributed with the VTK Visualization Toolkit [71]. The resolution is 256 x 256 x 256 voxels.

All data sets had previously been segmented; object voxels were set to value 1, and background voxels to value 0.

To create many different, yet similar objects, the input data was perturbed in a three step procedure. First, a skeleton was obtained from each volume, using the algorithm of W. Xie [87]. A graph was constructed from this skeleton; the vertices of the graph were the skeleton voxels, and edges were constructed between connected voxels. The graph was then simplified, and stripped of short end branches. The edges of this graph were subdivided into shorter edges, and the distance transform of the input volume was then sampled at each vertex of this graph. A root vertex was manually determined.

In the second step, the graph was perturbed, using a mass-spring system, with the edges modeled as springs, and small repelling forces acting between all vertices. A damping force allowed the system to settle over time. A number of randomly selected end and junction vertices, including the root vertex, was given a very large mass, to fix their position. Special constant forces were added between a number of randomly selected junction vertices, pulling them towards each other. All vertices, with the exception of the fixed vertices, were then given a random initial velocity, and the mass-spring system was subsequently left to relax over a number of iterations.

In the third and final step, the previously sampled distance transform at each graph vertex was randomly perturbed. These distances were then considered to be the radii of spheres, centered at each vertex. A volume was constructed around the graph, with all voxels inside these spheres set to 1, and all other voxels set to 0. To create a smoother object, the graph was first recursively subdivided so no edge was longer than the distance between two voxels. The distance data for the new points was interpolated. This volume was large enough to contain all the spheres.

3.4.5 Real World Objects Results

We view the results of the 15 measurements as coordinates in a morphological space. Determining the quality of a skeleton is then the same as determining the distance between the points defined by measurements performed on the ground truth and on the skeleton. The 15-dimensional morphological space is difficult to analyze. The ranges of the metrics are all different, and make a straight comparison impossible. In addition, a similarity factor consisting of 15 numbers is difficult to interpret.

To resolve these problems, we have chosen to first normalize the results, so they all range from 0 to 10. The range of the normalization is determined by all the results collected from all perturbations of a dataset. The next step is to calculate the error of each metric relative to the value measured using the ground truth. We have chosen to use the root mean

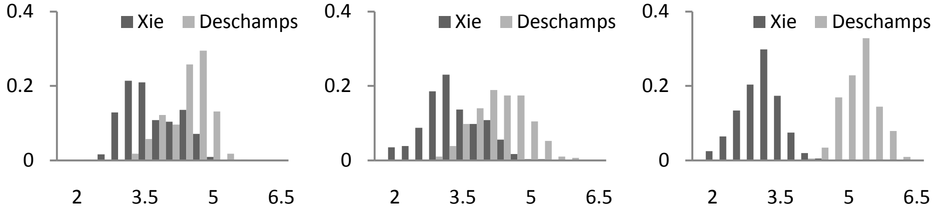


Figure 3.9: Error histograms for each input object. From left to right: coral, blood vessel, and pine root data. Along the x-axis is the RMS error, along the y-axis is the relative frequency.

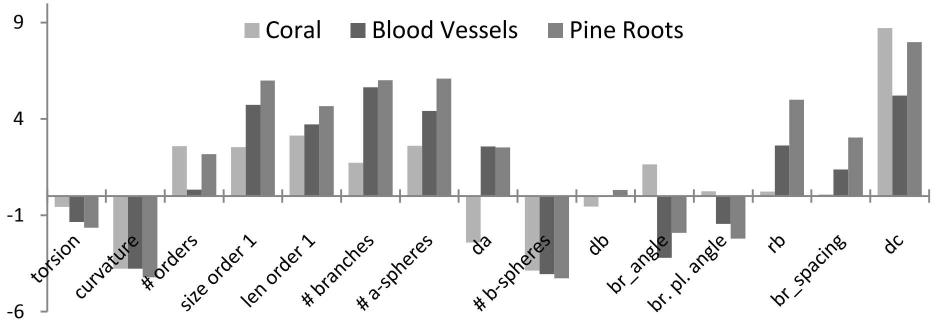


Figure 3.10: The relative error of the Xie algorithm. The error is shown subtracted from the error of the Deschamps algorithm, for each metric. Positive values indicate better performance than the Deschamps algorithm.

square of the errors along each dimension as the indicator of similarity. A choice for a different function, e.g., co-variance, can equally well be made.

Using this root mean square of the error we can now measure how well the extracted skeleton approaches the ground truth.

From each of the input data sets, we have created a large number of perturbed objects. Objects were discarded for which the generated volume topology did not match the original ground truth topology, for example because the spheres mapped onto different branches on the ground truth, creating additional loops. In total, 434 coral data sets, 286 blood vessel data sets, and 201 pine root data sets were used for the measurements.

The quantization factor used by the Deschamps algorithm was set to the maximum value of the distance transform for each object. Line simplification was performed with a threshold distance of 1.0, and end branches of the skeletons with a line length of less than 10.0 after simplification were discarded.

The graphs in Figure 3.9 show the distribution of the root mean square error in all dimensions for all the object perturbations along the x-axis, and the relative frequency on the y-axis. The error is relative to the ground truth, for both algorithms. An error value of 0 would mean that the results were identical to the ground truth. Separate graphs are shown for each original data set. The graphs clearly indicate that the skeletons produced by the

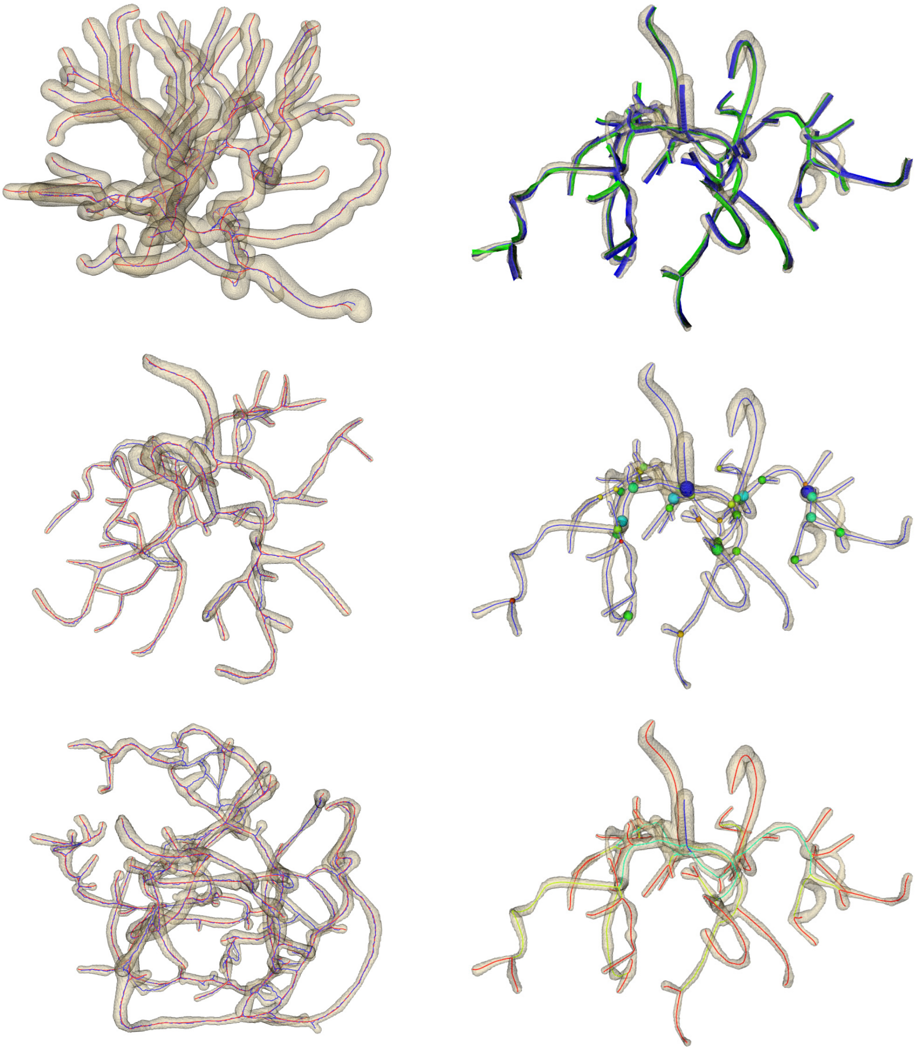


Figure 3.11: Sample permutations and measurements. The left column, from top to bottom, shows a permutation of the coral, blood vessel and pine root data sets, each with two skeletons. The right column shows measurements performed on the same blood vessel data shown in the middle image of the left column, using the Deschamps skeleton; from top to bottom, curvature and torsion, a-spheres and branch ordering (red is the lowest order, blue is the highest) are shown.

algorithm of Xie are overall better than those produced by the algorithm of Deschamps. The greatest difference can be found when looking at the pine root data set.

Because the fifteen metrics have been compiled into a single number, it is possible that the wave propagation algorithm produces better results for some metrics than the thinning algorithm. Indeed, according to the graph in Figure 3.10 this is the case. This graph shows, for each metric and dataset, the difference between the wave propagation and thinning algorithms. Positive values indicate better performance for the Xie algorithm, and negative values indicate a performance advantage for Deschamps' algorithm. The graph also shows that besides being dependent on the applied metric, the performance also depends on the data set being used.

The left column of Figure 3.11 shows a translucent iso-surface rendering of the volume constructed from a random perturbation of each input data set, along with the Xie (blue lines) and Deschamps (red lines) skeletons for that perturbation. It is difficult to evaluate the performance of the algorithms by visually comparing the two sets of lines.

The right column of Figure 3.11 shows three measurements performed on a wave propagation skeleton. The object is the same as the one in the middle of left column. In the top image ribbons were constructed by connecting the normal (green) and bi-normal (blue) vectors of the Frenet frames at each point of the skeleton; their twists show the curvature and torsion of the skeleton. The middle image shows the a-spheres in the image; the color corresponds to the radius of the sphere, with red for the smallest and blue for the largest radius. The bottom image shows the Horton-Strahler ordering of the skeleton, with order 1 in red, and the highest order in blue.

3.5 Conclusion

We have described the image processing steps needed to obtain a skeleton from a CT scan of a coral colony: noise filtering, segmentation and skeletonization.

The choice of noise filtering and segmentation algorithms was made from a number of available filters in the DeVIDE visualization and image processing environment, which is described in [11]. This environment provides convenient access to algorithms implemented in the Insight Segmentation and Registration Toolkit [34]. The actual algorithms were selected by visual evaluation of their output.

In order to reduce the number of noise-related artifacts in the data set, and based on the requirement of edge (and topology) preservation, anisotropic filtering is a suitable method. The CT scans were subjected to four iterations of the Curvature Flow noise reduction algorithm from ITK. The resulting data was then segmented with the ITK Confidence Connected algorithm, a region growing segmentation algorithm guided by simple statistics of an initial region around manually selected seed points. Finally our own hole-filling algorithm was applied to the resulting images to remove specific artifacts.

From the resulting binary images a three-dimensional morphological skeleton was extracted using the thinning skeletonization algorithm proposed in [87], which we have demonstrated to be very suitable for this purpose.

A comparison was made between skeletonization algorithms using thinning and wave propagation, by quantitative comparison of the skeletons using a variety of metrics for which the ground truth value was known. When looking at individual metrics, there is no clear winner. Sometimes thinning is better, and sometimes wave propagation. A subset of the metrics could be chosen such that the total score is better for each algorithm. This indicates that all algorithms have their strengths and weaknesses. The suitability of an algorithm is thus dependent both on the specific metrics that will be used with the skeleton, and on the data that will be analyzed.

When considering all the metrics, the thinning algorithm of Xie clearly outperforms the other algorithms. The difference between the algorithms also depends on the data used, showing some difference with coral, but a major difference when considering the pine root data.

Chapter 4 Measuring the Skeleton

This chapter deals with measuring an object using the skeleton. The first section explains how a graph is created from the skeleton for the purpose of measuring. The second section explains what measurements are used in the case of marine coral, and how these are implemented. The third section is concerned with the sources of uncertainty in the data. The fourth section describes the results of measuring three coral specimens.

4.1 Skeleton Graph

Most skeletonization algorithms produce output in the form of an image with the skeletal voxels marked. Such skeletons are not directly usable, for example they lack information about adjacency and connections between voxels. The voxels of the skeleton form the vertices of the graph G, but the edges E must be determined. Once G is obtained, this graph can be processed further. The image skeleton can be converted to a graph using the following method, as described in [65]:

- Skeleton voxels are added to the graph as vertices, with coordinates corresponding to the voxel location in the image grid.
- For each vertex v edges $vs_0 \dots vs_n$ are added where $s_0 \dots s_n$ are the neighboring skeleton voxels in the image. In order to avoid creating cycles at branching points in the graph, first only edges are added for vertices v with degree $d_G(v) \leq 2$. Edges for vertices with $d_G(v) > 2$ are added afterwards, avoiding the creation of edges vs_i if the graph already contains a path (v, S, s_i) where $S \subseteq \{s_0 \dots s_n\}$.

An example is shown in Figure 4.1. Here the boxes represent voxels, while the lines show the resulting graph.

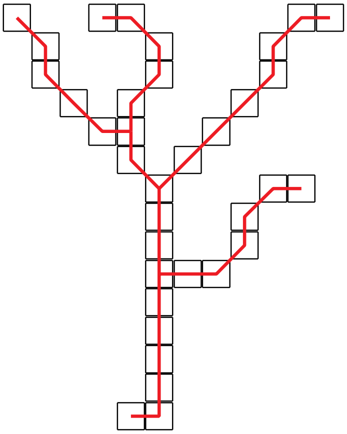


Figure 4.1: An example skeleton. The voxels are represented as boxes and the corresponding graph as a line.

The distance function $D_S(v)$ is then determined for each vertex v. For this, the Euclidean distance transform of the segmented image is used, as described in [10]; $D_S(v)$ is then the value of this transform at the voxel corresponding to each vertex v.

4.1.1 Graph Post-processing

Some post-processing is necessary on the skeleton graph before it can be measured. In part this serves to improve the graph, while other procedures are necessary to make it possible to perform the measurements at all.

- **Simplification**

As a result of the unevenness of the surface of an object, even after filtering, the skeleton may contain many ‘false branches’. These are branches that do not correspond to an actual branch of the coral, but rather to a small disturbance on the surface. While these branches could be removed manually from the skeleton, there can be a lot of such branches, and it is easier to remove them automatically. This can be achieved by removing each *terminal* branch b for which

$$\frac{|b|}{D_S(a)} < \varepsilon$$

where $D_S(a)$ is the distance function at the junction a of branch b, and ε is some constant that is determined empirically. The rationale is that such terminal branches are too short to be useful for measurement: if the length of a terminal branch on the graph is less than the thickness of the object at the start of that branch, then that branch does not even extend into the actual perturbation.

- **Straightening**

Because the skeleton is voxel-based, it will contain aliasing artifacts. These appear in the form of jagged lines that should have been straight. If this is considered to be a problem, for example because it increases the length of a branch, the lines can be straightened as outlined in [65].

- **Loop removal**

Some of the possible measurements require the branches to be ordered. This hierarchical ordering of branches is only possible if the skeleton graph contains no cycles, but unfortunately this is usually not the case. Loops can appear in skeletons either as an artifact of insufficient scanner resolution, or because the topology of the object contains actual loops (i.e., the branches of the specimen have actually fused together).

To properly perform the measurements, these cycles must be disconnected by removing at least one edge, and possibly several edges and vertices, from the graph. In some cases a complete branch must be removed. Automatic detection of the exact edges and vertices that must be removed to perform such disconnections is not currently possible. However, the number of necessary manual edits is very low (2 to 20 in our test objects), and the task can be quickly performed even by inexperienced users. The system can detect and highlight the cycles in the graph, and the user can then indicate which parts of the cycle should be removed.

- **Ordering**

A hierarchical Horton-Strahler ordering is applied to the graph. The Horton-Strahler ordering is a topological ordering of the branches of a tree, originally used in the analysis of river networks [33,76,88]. This ordering is applied to the branches B as follows:

First, the graph G must be a tree. A root vertex r is selected. Directionality is added to the graph, with all edges pointing away from r. All terminal branches are assigned order 1. Then for each branch for which order ω is unknown, ω is calculated from the orders ω_1 and ω_2 of its two child branches as:

$$\omega = \max(\omega_1, \omega_2) + \delta(\omega_1, \omega_2)$$

where

$$\delta(\omega_1, \omega_2) = \begin{cases} 1, & \omega_1 = \omega_2 \\ 0, & \omega_1 \neq \omega_2 \end{cases}.$$

A branch can thus only be assigned an order if the orders of its child branches are known. The process iterates until all branches have been assigned an order.

- **Metadata addition**

It may also be necessary to add certain metadata to the graph to enable measurements, unless this data can be determined automatically.

For tree-like structures such as coral colonies, it is necessary to know where the root of this tree is located in the graph. For some objects this can be determined automatically, but in other cases this information must be added to the graph manually.

It may be necessary to determine the orientation of an object in its original environment. It is quite likely that an object is not being scanned in a perfect upright orientation, instead being placed inside the scanner in a convenient, stable orientation that is completely unrelated. This is especially true for objects that have been attached to something and do not have a wide base on which they can stand. If the original orientation of an object is necessary and cannot be determined automatically, the rotation between the image and the original orientation has to be applied manually.

This information can be applied to the skeleton graph either by applying the rotation transformation to all the vertices, or by storing the rotation and applying it to any relevant metrics. The former method may be useful to properly visualize the data, both the image and the skeletal graph.

4.2 Metrics

The skeleton graph provides convenient access to various topological and geometrical features of the skeleton and thus of the object. It is possible to define a great variety of metrics that make use of this graph, either directly or by using it to select specific parts of the object.

4.2.1 Coral Metrics

Here we list the metrics used in coral biology, along with how these are implemented. Originally these were developed for 2D images [1,40,70,73], but they have been extended for use in 3D analysis [41].

- **Maximum thickness**

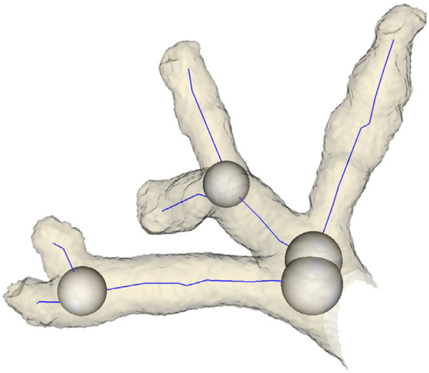


Figure 4.2: Maximum thickness *da* (a-sphere).

The maximum thickness da of a branch is defined as the thickness of the branch at a junction of the skeleton graph. It is the radius of the maximum inscribed sphere, centered at the junction, touching the background in the original voxel image. This radius is thus equal to the value of the distance function $D_S(v)$ at the junction point, and can thus be directly obtained as $D_S(v)$ for all $v \in V(G)$ for which

$$d_G(v) > 2.$$

It is thought that the branches of the coral grow to the highest thickness just before they split into new branches. This thickness is largely dependent on the species of coral, but often also a dependency on environmental parameters can be observed.

These spheres are referred to as a-spheres and are depicted in Figure 4.2.

- **Minimum thickness**

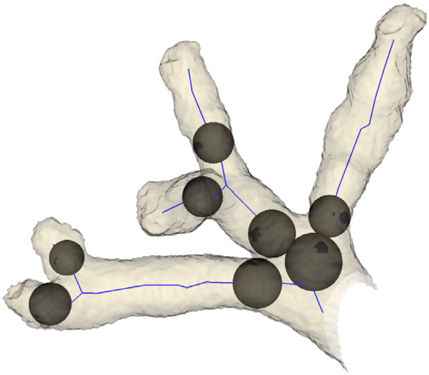


Figure 4.3: Minimum thickness *db* (b-sphere).

The minimum thickness db of a branch is the radius of a sphere located on the part of the skeleton graph branch following a junction, with a radius such that it touches both the coral surface and the a-sphere at the junction. These spheres are referred to as b-spheres. The method to locate the b-spheres is to consider the distance function $D_S(v)$ along the skeleton graph following a junction, and selecting the best matching location. When considering the spheres this means locating

$$\min_{\mathrm{b}} \left| \|a - b\| - (r(a) + r(b)) \right|$$

where a, b are the locations of the respective spheres, and $r(x) = D_S(x)$ is the radius of a sphere. A topological ordering of the graph is necessary to perform this measurement, as these spheres are always only located at the start of a branch.

As with the maximum thickness, a large variation can be observed between coral species and smaller variation exists between colonies of the same species.

Figure 4.3 shows the b-spheres.

- **Terminal thickness**

The terminal thickness dc of a branch can only be defined for terminal branches. It is the radius of a sphere located at or near the tip of a terminal branch of the skeleton graph. This metric is rather dependent on the location in which any particular skeletonization algorithm might terminate branches. If the algorithm produces long branches that terminate close to the border of the object, the resulting values of dc are too low. This can be mitigated by locating a local maximum on the branch, i.e., a vertex v_i on the branch $v_0 \dots v_n$ for which

$$\begin{aligned} D_S(v_0 \dots v_{i-1}) < D_S(v_i) \\ D_S(v_{i+1}) \leq D_S(v_i) \end{aligned}.$$

However, if the algorithm produces branches that are too short, this method has no effect. Figure 4.4 shows this metric.

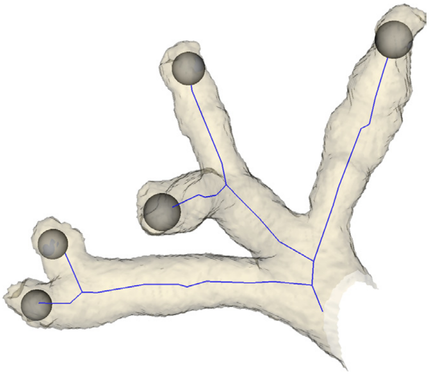


Figure 4.4: Terminal thickness *dc*.

- **Branching rate**

The branching rate rb is defined by the distance between two successive junction points in the graph, i.e., $|v_0 v_n|$ for a branch. It is a measure of how often new branches are formed; a high value indicates relatively slow formation of new branches during the growth process as branches tend to grow without splitting, while a low value indicates that branches form in quick succession. There is an implicit assumption that the growth speed of the coral is constant.

Figure 4.5 shows the branching rate as red lines.

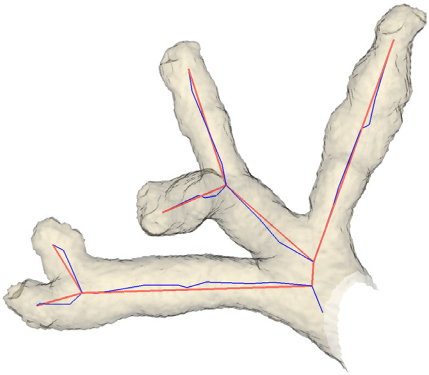


Figure 4.5: Branching rate shown in red.

- **Spacing**

The branch spacing *br_spacing* is defined as the distance between the endpoint of a branch, and the closest location on the graph which does not belong to the current branch. The method used to find the spacing is to intersect the graph with a sphere with radius r centered at the terminal vertex t of the branch. The subgraph G_t is composed from the vertices $v \in G$ for which

$$|\overrightarrow{tv}| \leq r$$

along with the edges which connect these vertices in G. The spacing is then defined as the smallest possible r for which G_t is an unconnected graph.

This standard deviation of this metric expresses the degree to which branches will tend to fuse; a relatively low value indicates that there is a low degree of self-intersection,

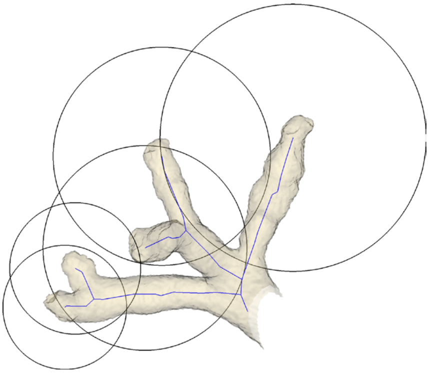


Figure 4.6: Branch spacing.

while a high value shows the lack of a mechanism which would prevent self-intersection.

Figure 4.6 shows the branch spacing as the outlines of the clipping spheres.

- **Geotropy angle**

The geotropy angle *g_angle* is the angle between the normal of a predefined ground plane, and the vector $\overrightarrow{v_0 v_n}$ between the start vertex v_0 of a branch (i.e., the center of the *a*-sphere of the parent branch) and the endpoint v_n of the same branch.

The branches of many coral species show negative geotropism, i.e., they grow away from the substrate. This is the result of the presence of symbiotic photosynthetic algae, which produce more nutrients if the coral receives more light, thus favoring growth towards light.

It can be seen as the angle between the solid red and dotted green lines in Figure 4.7.

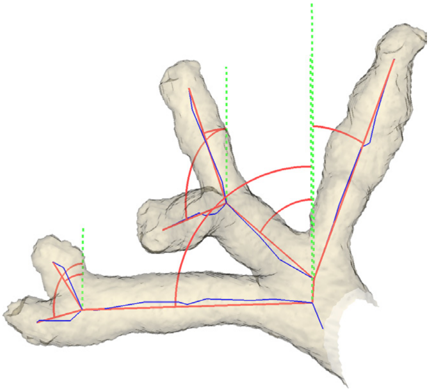


Figure 4.7: Geotropy angles.

- **Branching angle**

Given a non-terminal branch, the branching angle *b_angle* is the angle measured between the two vectors $\overrightarrow{ab_1}$ and $\overrightarrow{ab_2}$, where a is the center of the *a*-sphere of a branch, while b_1 and b_2 are the centers of the *b*-spheres of the child branches. The value of this metric is believed to be characteristic for a species, thus constant or at least very similar between specimens.

This is depicted in Figure 4.8.

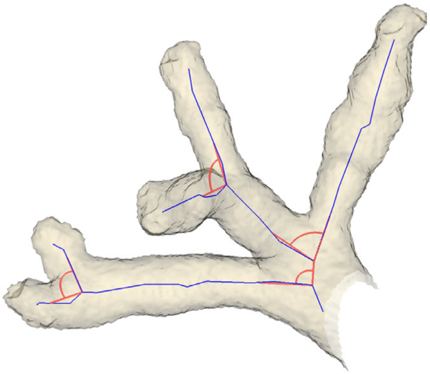


Figure 4.8: Branching angles.

4.3 Accuracy and Uncertainty

It is important to remember that measurements do not have infinite accuracy. Uncertainty reduces the accuracy at every step of the measuring process, starting even before an object is scanned, and growing with each consecutive step. Any inaccuracy in the process will result in an inaccuracy of the measurements.

We discuss here some sources of uncertainty, as it is important to know where the uncertainty originates. The most well-known uncertainty is from the method used to obtain an image, but there are other sources as well.

4.3.1 Object Noise

Object noise refers to various artifacts present in the object itself. Although they are present in the object, they are not considered to be part of it. Since the focus is on measuring an 'idealized' version of the object, such artifacts are undesirable.

We discuss here a number of sources of noise specific in (CT scans of) marine corals.

- **Surface**

Corals often tend to have an uneven surface. This might be a natural property of the species and can be related to the morphology of the individual corallites. The remains of corallites can be clearly seen on the surface, and due to resolution limits show up in CT scans as lower density areas.

The rugosity can be further influenced due to erosion processes resulting from long-term exposure to the marine environment. In *M. mirabilis* colonies the polyps at the bases of branches usually die as the branches grow, exposing the skeleton.

- **Ongrowth**

The surface might be partially covered by encrusting animals or algae. This is different from the regular uneven surface of corals, as these artifacts do not constitute part of the actual coral. However, they are part of the collected specimen, and appear in the scan. Some of this additional matter has a different density than coral, thus is already excluded during segmentation. Some animals however, such as for example tube worms, deposit calcifications on the coral that are indistinguishable in their density.

- **Predation**

Coral serves as a food source for other organisms. Larger creatures may take bites out of branches, or consume whole branches. Smaller organisms, such as various worms, live inside the coral and eat their way through the branches.

- **Damage in habitat, and during/after collection**

Being rather fragile, coral branches can break off. Such damage may have occurred already in the natural habitat, or it may happen during collection of a sample. Also, after being collected the coral becomes even more fragile, thus broken branches occur frequently. These are often glued back to the specimen, and the glue shows as a different density on a scan.

4.3.2 Processing Artifacts

Uncertainty can be introduced and propagated in various processing steps. Noise filtering and segmentation of the images both introduce some uncertainty, as they affect the shape of the object.

Skeletonizations aims to detect features of an object, however there are many ways in which this can yield invalid or uncertain results. As it is the skeleton that is measured, this directly affects the accuracy of the measurements.

- **Spurious branches**

Branches appear in the skeleton where no such features are present in the object. This is a result of the sensitivity of the algorithm to small surface perturbations. It is a property of the algorithm, but in general it is more likely to occur as the thickness of the object, as measured in voxels, increases, either due to increased resolution or actual object size.

- **Missed branches**

A branch is not present in a skeleton where the object does in fact have a branch. This is the complement of spurious branches, as it is caused by reduced algorithm sensitivity to surface perturbations. It is similarly also more likely to occur in thinner objects.

- **Location**

Although the skeleton is supposed to be at the center of an object, it may in fact be located slightly off-center. This is especially true for skeletons of objects that have a thickness of an even number of voxels, in which case the skeleton will always be off by at least one half voxel. It then depends on the algorithm and on the orientation of the object on which side of the true center the skeleton will be located.

- **Branching point location**

The location of branching points in a skeleton also differs between algorithms, especially when more than two new branches originate in the same location. Depending on exact circumstances, it is possible that in such a location an algorithm will create a single branching point with multiple branches, or, more likely, multiple branching points very close together. In the latter case it is also possible that the order in which the branches start from the main branch differs. In either case the topology of the resulting skeleton will not exactly match the topology of the original object.

- **Terminal branch length**

The length of terminal branches in a skeleton is extremely dependent on the algorithm. Some algorithms yield terminal branches that continue all the way to the boundary of the object, while others, particularly thinning algorithms, create short to very short branches that terminate far from the boundary.

- **Measurement implementation**

The measurement methods themselves are another source of uncertainty. Sometimes the metric is not a directly measurable feature in the data but rather some sort of best-fit approximation is used. There might also be a small tolerance involved in locating the site of a feature. Such metrics have an inherently larger uncertainty than those that are measured directly.

4.4 Application

We have performed measurements of samples of the branching scleractinian coral *Madracis mirabilis* collected at different depths. We demonstrate that the morphological variation of these samples can be quantified, and that biologically relevant morphological characteristics, such as the metrics in section 4.2.1 and surface/volume ratios, can be computed.

4.4.1 Measurement Methods

CT scans of the corals were subjected to four iterations of the curvature flow anisotropic noise filter. The filtered images were then segmented using the confidence connected region growing segmentation algorithm with manually selected seed points. The segmentations were post-processed using our hole-filling algorithm. These steps were explained in chapter 3.

From these segmented images we extracted the morphological skeleton using the thinning algorithm of W. Xie, which was discussed and evaluated in chapter 3. The skeleton was subsequently converted into a graph representation. The next step was the removal of loops from the skeleton. Loops were detected automatically in the branch ordering process, and user interaction was used to indicate the location and the size of the necessary disconnection in the skeleton graph. Finally, to enable correct measurements, the root of the coral was selected manually and the proper orientation of the coral with respect to the ground was established interactively. After this the graph was used to measure the coral colony using the metrics described in section 4.2.1.

The distribution of the measurements was investigated by applying the non-parametric Mann-Whitney U two-sample rank test [51]. In all cases the hypothesis that the distribution of measurements of both objects is the same is tested against the alternative that they differ by a translation.

4.4.2 Results

The variation in numerous measures of colony morphology of M. mirabilis detected by the analysis procedure was consistent with previously observed trends of morphological variation for this species. The largest branch spacing (*br_spacing*) was found in the thin-branching deep-water morphology, and the branch spacing of the samples increased with depth (Table 4.1, Figure 4.14). Across the same depth range the standard deviation of the branch spacing also decreased, indicating a more regular branching pattern (Table 4.1, Figure 4.14). For the sample from deep water, there were no values close to zero for this parameter, which demonstrates that no anastomosis was present in this colony (something that could be verified in the loop removal procedure discussed in the methods), while for the intermediate and shallow depth samples, values near zero occurred and anastomosis of branches did occur. When the standard deviations of the measurements of the geotropy angles (*b_angle*) and the branching angles (*b_angle*) were compared (Table 4.1), it was observed that with decreasing depth branches tended to form in almost all directions, while in deep locations most branches would be formed in vertical directions. The branching rate (*rb*) increased with decreasing depth (Table 4.1, Figure 4.14). The more compact shallow and intermediate depth morphs were characterized by a relatively higher degree of branch formation.. The measurements on the minimal (*db*), maximal (*da*), and terminal thickness of branches (*dc*) all showed a similar tendency (Table 4.1, Figure 4.14) where the thickness increased in the series deep morph, shallow morph, intermediate morph. The object from deep water was characterized by a clearly lower value of *da*, *db* and *dc* and was clearly a thin-branching growth form when compared to the compact forms from lower depths. In all thickness measurements, the thickness of the intermediate morph was significantly characterized by a slightly larger mean thickness compared to the shallow water morph. In all thickness measurements the mean value of *dc* was significantly below the mean values of *da* and *db* (Table 4.1). All morphological measurements have also been compared using a nonparametric two-sample rank test (Table 4.2). The surface area and volume of each sample were measured, and the surface to volume ratio exhibited the opposite tendency of the branch thickness measurements, decreasing from the deep water sample to the intermediate depth sample (Table 4.3).

The distributions of the measurements are also shown separately for each branch order in Figure 4.10 through Figure 4.13. A 3D visualization of the maximum branch thickness for the deep water morph is shown in Figure 4.9.

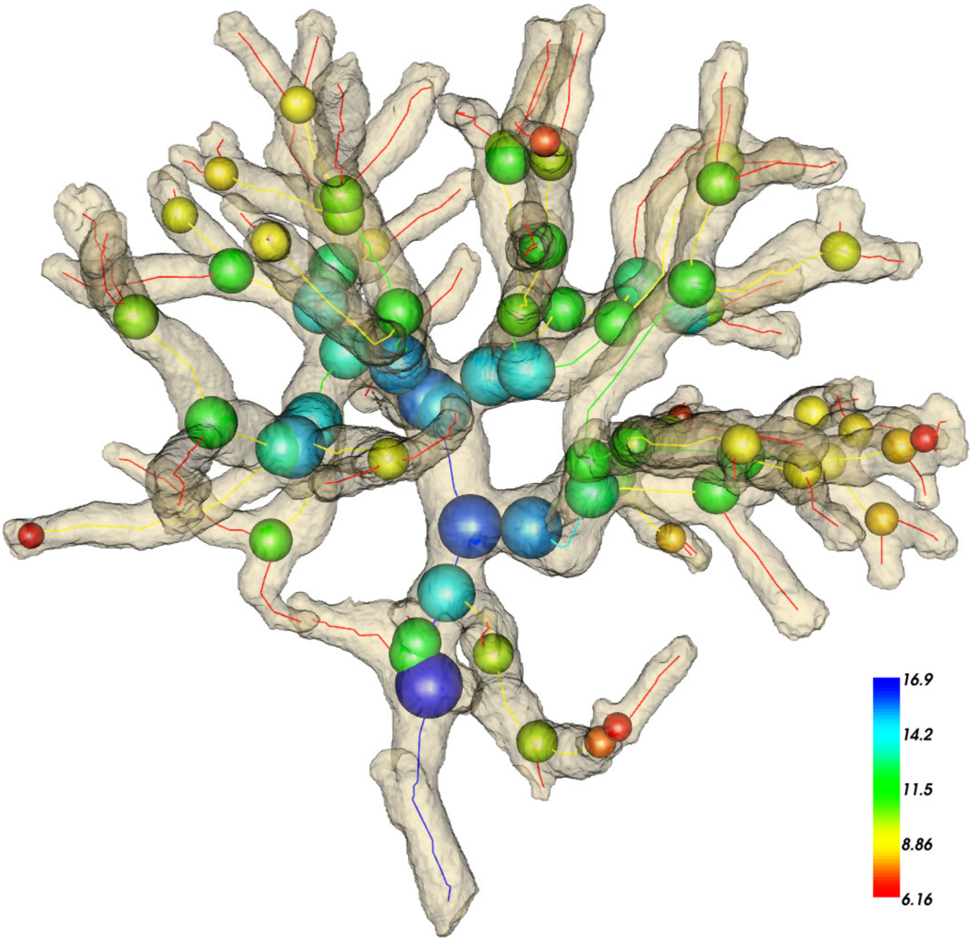


Figure 4.9: A 3D visualization of the measured *a*-sphere diameters. The maximum branch thickness is shown here for the deep water coral as spheres. The coloring of the spheres indicates diameter; the coloring of the skeleton graph indicates the branch order.

measurement	**dataset**	**samples**	**median**	**mean**	**std**	**skew**	**kurtosis**
maximum thickness (da)	shallow	172	6.8	6.8	1.2	-0.08	-0.25
	middle	175	7.3	7.4	1.4	0.02	-0.22
	deep	65	5.5	5.7	1.2	0.49	-0.62
minimum thickness (db)	shallow	345	5.4	5.5	1.6	-0,22	0.47
	middle	351	5.9	5.9	1.9	-0,51	0.42
	deep	131	4.4	4.7	1.3	0.82	0.49
end thickness (dc)	shallow	173	4.2	4.1	1.3	-1.30	1.73
	middle	176	4.5	4.2	1.7	-0.45	0.03
	deep	66	3.2	3.2	0.8	-0.29	-0.39
branch spacing (br_spacing)	shallow	172	8.4	8.5	2.1	-0.16	-0.14
	middle	175	9.9	9.9	3.1	0.10	-0,.74
	deep	65	11.6	11.8	4.2	0.40	-0.28
branching angle (b_angle)	shallow	172	88.8	90.9	21.9	0.21	0.11
	middle	175	84.6	86.2	16.9	0.20	0.21
	deep	65	89.4	91.0	13.6	0.29	-0.07
geotropy angle (g_angle)	shallow	345	61.5	70.0	36.2	0.57	-0.30
	middle	341	52.6	61.8	34.2	0.82	0.19
	deep	131	70.4	71.5	33.8	0.53	0.34
branching rate (rb)	shallow	345	7.3	8.2	5.7	1.34	2.54
	middle	341	9.9	11.0	7.9	1,059	1.30
	deep	131	12.3	13.1	7.2	0,761	0.36

Table 4.1: Results of the measurements. All thickness and distance measurements are in mm, all angles are in degrees.

measurement	**shallow and middle**	**shallow and deep**	**middle and deep**
maximum thickness (da)	<	>	>
minimum thickness (db)	<	>	>
end thickness (dc)	0	>	>
branch spacing (br_spacing)	<	<	<
branching angle (b_angle)	>	0	<
geotropy angle (g_angle)	>	0	<
branching rate (br)	<	<	0

Table 4.2: Results of a two-sample rank test for the distributions of the measurements of two objects. With '<' is indicated that the distribution of object 1 is located left to the one of object 2; '0' indicates that the distribution of measurements of both objects is the same and '>' indicates that the distribution of object 1 is located right to the one of object 2.

object	**volume segmentation (cm^3)**	**volume iso-surface (cm^3)**	**area iso-surface (cm^2)**	**surface / volume ratio**
shallow	104.1	97.8	59.8	0.61
middle	185.1	173.0	100.2	0.57
deep	44.2	40.9	31.3	0.77

Table 4.3: Surface (cm^2), volume (cm^3), and surface / volume ratios of the samples.

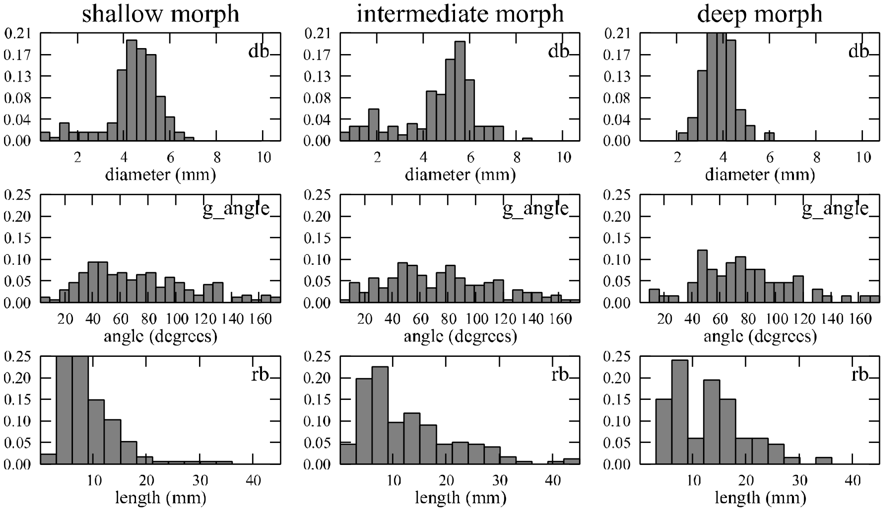


Figure 4.10: Histograms for branches of order 1 (terminal branches).

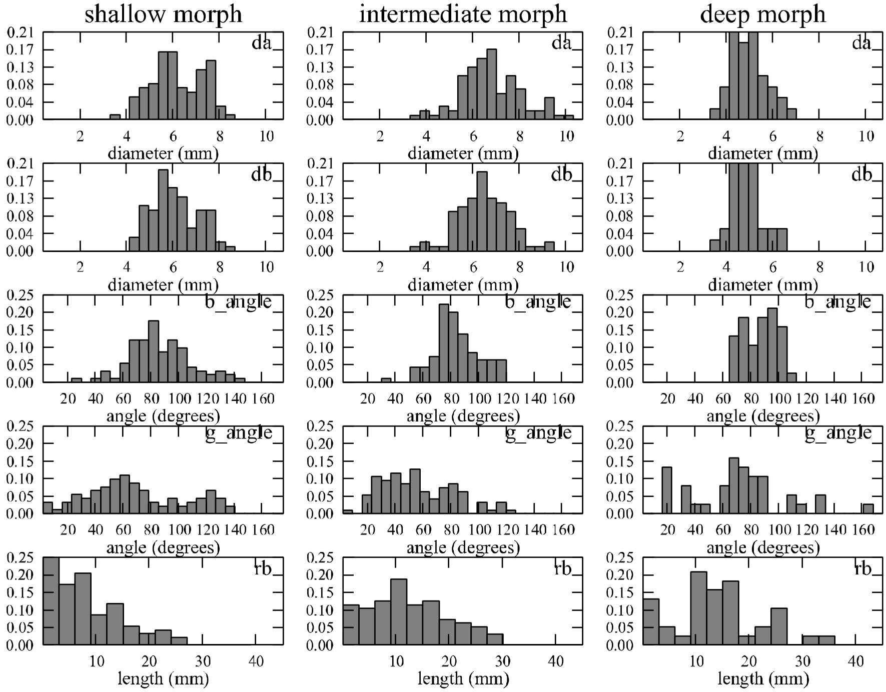


Figure 4.11: Histograms for branches of order 2.

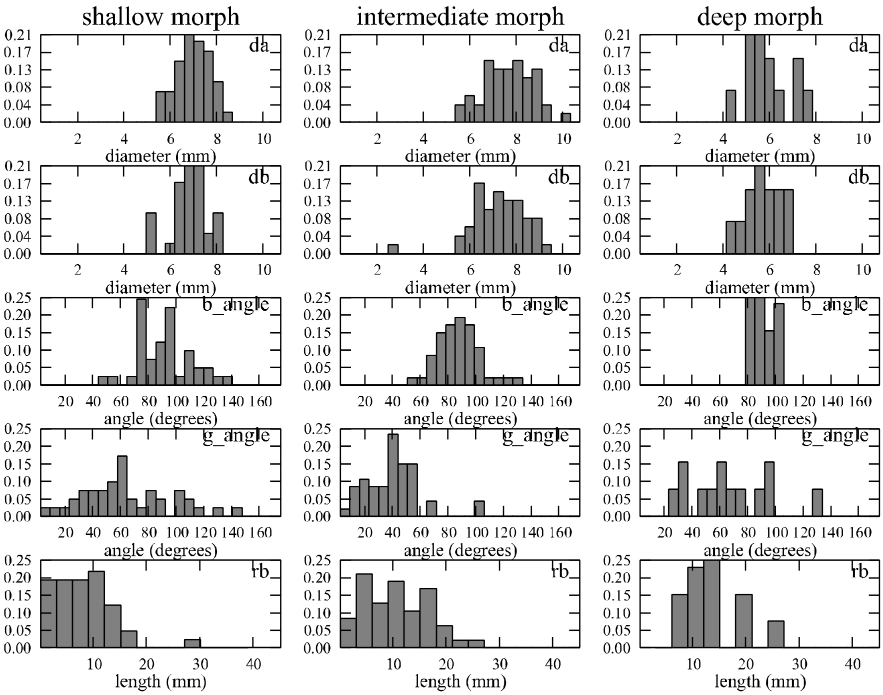


Figure 4.12: Histograms for branches of order 3.

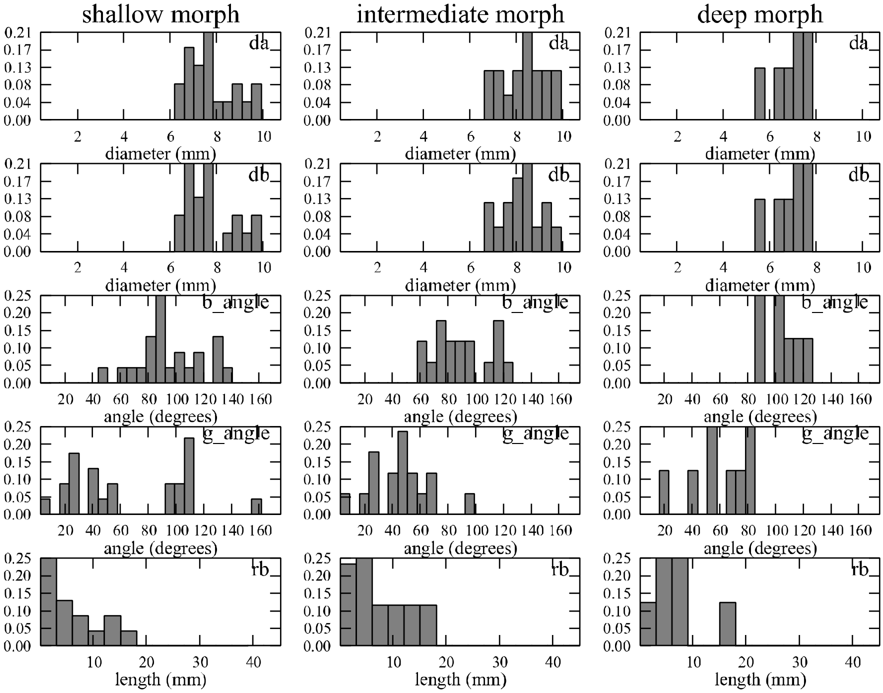


Figure 4.13: Histograms for branches of order 4.

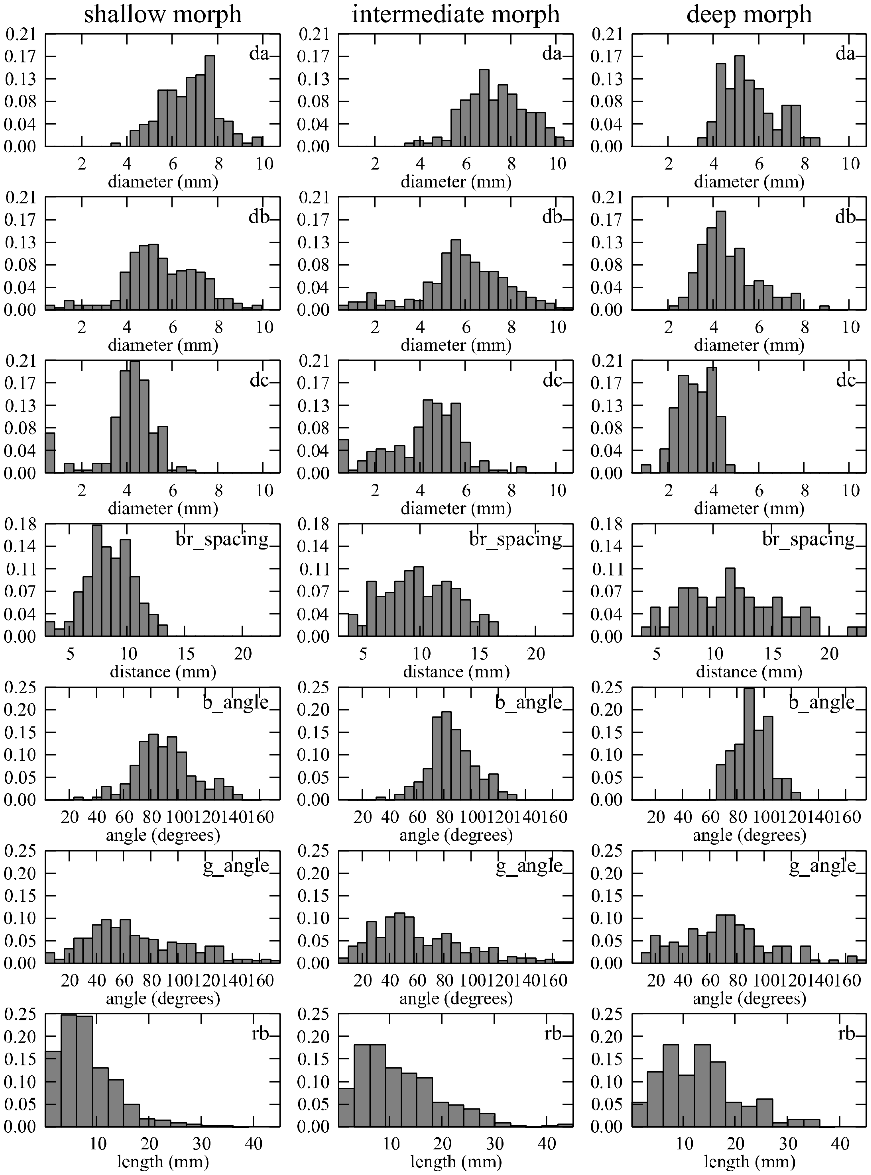


Figure 4.14: Histograms for the measurements. Diameter of maximum, minimum and terminal thickness (*da*, *db* and *dc*), the branch spacing (*br_spacing*), branching angle (*b_angle*), geotropy angle (*g_angle*) and branching rate (*rb*). The histograms show the values of the measurements on the x-axis, and the frequency distribution of these values on the y-axis. From top to bottom, each set of histograms shows the distribution the measurements of *da*, *db*, *dc*, *br_spacing*, *b_angle*, *g_angle*, and *rb* for respectively the colonies from shallow, intermediate depth and deep water. All thickness and distance measurements *da*, *db*, *dc*, *br_spacing* and *rb* are in mm, all angles *b_angle* and *g_angle* are in degrees.

4.5 Conclusion

This chapter demonstrates that it is possible to analyze indeterminate growth forms in volumetric data sets stemming from an X-ray Computed Tomography scanner by constructing a morphological skeleton. An important precondition for constructing a morphological skeleton is that the data set is preprocessed in such a way that all artifacts (for example cavities, holes in the skeleton, scanning artifacts, etc.) have been removed. For the morphologically variable coral *M. mirabilis*, these artifacts can be removed without compromising the major morphological characteristics of the colony. The analysis critically depends on the availability of morphological skeletons. Without these skeletons it is not possible to determine the various local morphological parameters (angles, thickness of branches, branching rates) in a useful way. Only a limited set of morphological characteristics has been quantified. By using the morphological skeleton, there is a straightforward way to simplify the morphology (while preserving the topology) of the colony and a large variety of (local) morphological characteristics can be defined. The morphological skeleton can be used in a visual inspection of the data set, something that is required in many of the data processing steps, and the hierarchical structure of the skeleton can be used in various processing algorithms.

In addition to the local morphological measurements, the computed morphological skeleton, the surface and the volume of the three-dimensional image of the coral can be used to estimate the values of a series of global measurements: fractal dimensions characterizing the rugosity of the surface, the space-filling properties of the coral surface, space-filling properties of the morphological skeleton of the object, surface-volume ratios or the degree of compactness of the colony (see also [43]). In general, the local morphological characteristics, as for example *br_spacing*, can be more easily linked to biological properties of the objects, while global characteristics lump together many different morphological properties into one number. Potentially the surface-volume estimations can be used to calibrate field methods available for three-dimensional morphometrics of complex-shaped marine sessile organisms [13,15].

This analysis demonstrates that it is possible to quantify a range of morphological characteristics in the samples, to construct a morphological “fingerprint” of a certain morph, and to compare it to other growth forms. In all thickness measurements the mean value of *dc* was significantly below the mean values of *da* and *db*, although it would be expected that *dc* is located between the mean values of *da* and *db*. In a branching coral without tapering ends the thickness of the branches will be the largest at the point of branching and the smallest immediately after branching. Here it should be noted that the value of *dc* strongly depends on the skeletonization algorithm and the determination of the endpoints. Another explanation here is that the tips of the colony are initially somewhat tapered in the growth process. In the deep water sample shown in Fig. 3c the tips of the colony exhibit some tapering. The degree of regularity of a branching pattern can be determined by considering the variation of the branch spacing (*br_spacing*) and the branching rate (*rb*), with low variation indicating a regular pattern. Furthermore, the values of *br_spacing* indicate if there is any fusing of branches in the sample. For example for the deep water object there are no values close to zero, which demonstrates that there is no anastomosis, which is an important biological property. It is possible to detect and quantify branch fusing, but there is no automated method for detecting the exact location of anastomosis. This analysis method requires the detected fusing to be manually removed. For organisms with a high degree of anastomosis this might increase the time taken to complete sample analysis. The variation of the geotropy angles (*g_angle*) quantifies if branches tend to form in almost all directions. Quite remarkable is that the mean value of the branching angle (*b_angle*) is nearly similar in the three objects, which corresponds to an earlier observation [40] that *b_angle* might be a morphological invariant in branching corals. The standard deviation of *b_angle* decreased with increasing sample depth, which demonstrates that *b_angle* is probably in practice not very useful as a morphological invariant characterizing for example a certain species and that there may be a physical / biomechanical explanation for this phenomenon.

Chapter 5 Visualization of the Measurements

Once the coral measurements are acquired, the results can be analyzed. One method of analysis is to use statistics and consider the resulting numbers. But often to gain a better understanding of the data various visualizations can be used. This also makes it much easier to spot interesting or suspicious patterns in the data.

Measurement results can be visualized either as statistical data plots, which give a good overview of the range and distribution of the values, or as visual representations of results values in the context of the original object, which can be used to determine the measured values at specific locations.

The advantages of both 2D plots and 3D visualizations can be combined through interaction to create a powerful analysis tool. The two different views are linked and interactive selection of part of the visualized data, also called brushing, can be performed on one view to highlight the same data in the other view. This type of interactive exploration makes it very easy to observe what values occur in a particular region of the coral, or conversely in which locations specific values occur.

A particular use for this is determining whether outliers or other unexpected results are the effect of artifacts in the data, or whether these results are actual valid features of the coral. In simple objects artifacts are usually easy to spot when examining a visualization of the object or the measurements. Very large and obvious artifacts are in general also easy to find in complex objects such as corals, but smaller artifacts can easily be overlooked. Since artifacts affect the results of the analysis, they can be found by locating unusual patterns in statistical plots of these results and investigating where in the coral these patterns originated. If necessary, they can then be edited out of the data and the analysis can focus on the remaining valid measurements.

An important but often overlooked aspect of analysis is the uncertainty of the data, as this determines the reliability and accuracy of the conclusions drawn from analysis. Standard methods exist for showing error and uncertainty in 2D plots, there is however much less consensus on how to achieve the same in 3D scientific visualizations.

The first section of this chapter explains how interactive visualizations are used throughout the measurement pipeline. The second section discusses how interaction and linked 2D and 3D visualizations can be used to explore the data. In the third section it is explained how uncertainty in the data can be visualized.

5.1 Editing

Interaction and visualization is used throughout the measurement pipeline, as indicated by the gray arrows in Figure 1.1 in the introduction (Chapter 1). An important part are the interactive visualizations, which are used when editing or inspecting the data during the steps leading up to the measuring:

- Qualitative assessment of the result of noise filtering.
  To assess the effect of a particular noise filtering algorithm, and to determine optimal parameters, a visual comparison is made between filtered and unfiltered data. Both 2D slices of the 3D data and 3D volume rendering are used for this purpose.
- Placement of the seed points used by the segmentation algorithm.
  The points are placed using arbitrarily oriented 2D slice planes. This is augmented with a translucent iso-surface of the data extracted at an iso-value that gives a clear indication of the three-dimensional structure of the coral, providing a useful context for the slices.
- Selection of correct fills as produced by the hole filling algorithm.
  A solid, colored surface of a proposed hole fill is shown together with a translucent surface rendering of the surrounding original segmented data. Using this, the user can decide whether the filled area actually corresponds to a hole, or that the algorithm mistakenly filled an area outside the coral, in between the tightly packed branches.
- Disconnecting loops in the skeleton graph.
  As loops in the skeleton can be detected, but not automatically disconnected, the user is presented with the skeleton with the loops highlighted. The user can then disconnect these loops by removing parts of the skeleton graph. See also Figure 5.1.

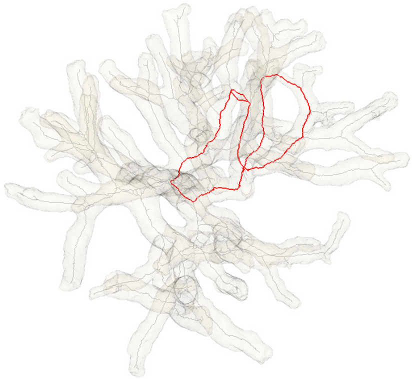


Figure 5.1: Highlighted loops in the skeleton.

- Interactive selection of the parameters for the simplification and straightening of the skeleton graph.
  The simplification and straightening of the graph is controlled by parameters that can be set by the user. These parameters can be interactively adjusted, and the resulting changes to the skeleton graph are immediately highlighted so that they can be compared to the original graph.
- Adding metadata to the skeleton graph: selection of the root branch and adjusting the substrate orientation.
  This information also makes use of interactive visualizations. To select the root, all

potential root locations on terminal branches are marked with small spheres. The current root is highlighted, and the other spheres can be clicked to select a new root. The substrate orientation is adjusted by interactively orienting a plane; the normal vector of this plane can also be entered numerically.

5.2 Exploring Measurements

Interactive visualizations make it possible to reach insights and make discoveries that would be very difficult, if not impossible, to achieve using traditional static visualizations and mental processes.

Brushing is a collection of techniques for selection and dynamic querying of values on a display, typically on statistical plots and other information and data visualization techniques. It was formally defined by Becker and Cleveland [5]. It is a technique often used to mark multivariate data points in linked views. Data that forms an interesting pattern in one view is brushed and can instantly be inspected in another view, possibly even across a network on a different computer [3]. This can be especially useful to visually explore huge datasets [44]. Roberts and Wright [67] have extended the concept of brushing to include not just data views but also various kinds of meta-information displayed along with the data.

Brushing has also been used to explore spatial measurement data, combining and linking statistical 2D views and spatial 3D views of the data [28], however this work is focused on exploring spatial distribution of statistical patterns using data from multiple sources.

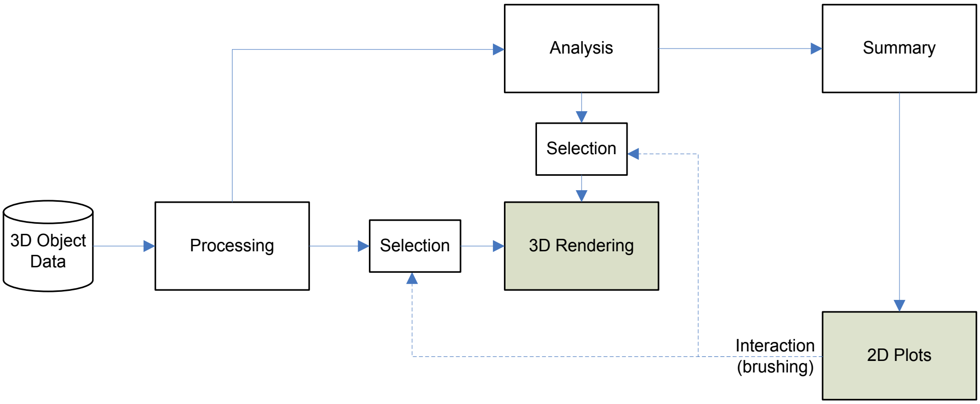


Figure 5.2: An overview of the exploration system. Interaction with 2D plots is used to modify the 3D view.

We have created a system that can be used to analyze and explore artifacts in the coral data. The measurement results can be summarized in statistical plots. The measurements are also shown using appropriately chosen geometric representations of the measured values (for example spheres to indicate diameters) in a 3D visualization, which also displays the original coral and the morphological skeleton. The plots can be brushed, and this brushed selection is used to pick only the data points corresponding to the selection for visualization in the 3D view. This greatly reduces the complexity of the visualization, and it allows easy inspection of disputed measured values within their original context, and thus to assess their validity. A diagram of the system is shown in Figure 5.2.

The plots are typically either histograms of individual metrics, or scatter plots of any combination of different metrics. In histograms a value range can be brushed by drawing a line on the plot using the mouse. If the bars of the histogram are wider than a single pixel the whole range of a bar is selected if the line covers any part of that bar. The lower value of the leftmost selected bar and the upper value of the rightmost selected bar are then used as threshold value, and measurements falling within the threshold range are selected for the 3D visualization. A selection made on a histogram is shown in Figure 5.3.

In the scatter plots a lasso tool is used to select points. Using the mouse a contour can be drawn on the plot, and upon release of the mouse button this contour is closed. The system then determines which plotted points are inside the contour, and which original measurement points correspond to the selection. These points are then selected for the 3D visualization. This can be seen in Figure 5.6.

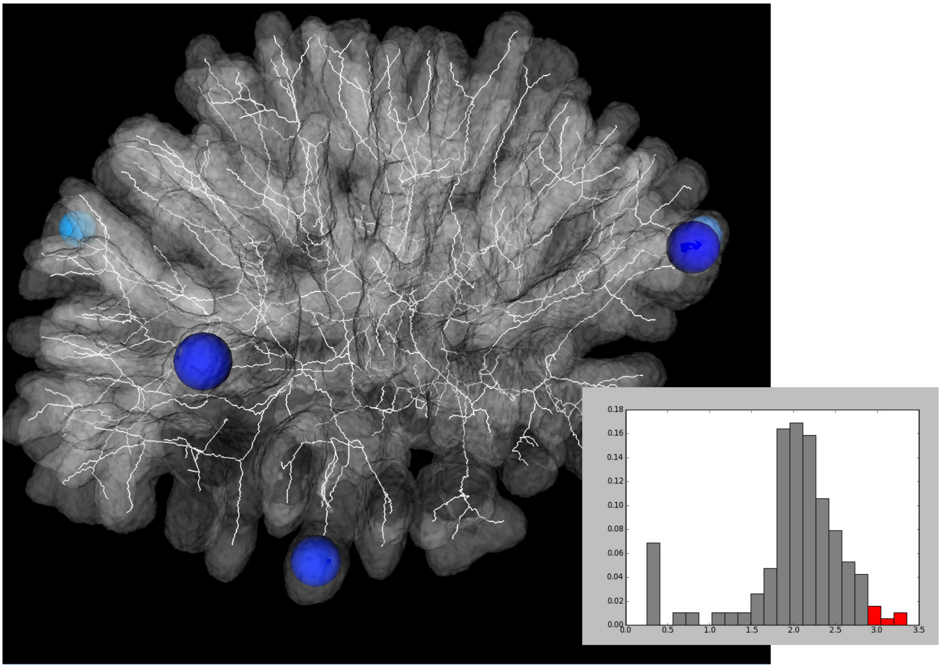


Figure 5.3: Selection of measurements using a histogram. A small number of measurements at the high end of the range are selected.

In case a different kind of plot is deemed more useful to display information about a particular type of data, the interaction is trivial to add to any other type of plot, for example statistical box plots or line plots. It is also possible to make more than one selection in a plot, and to combine selections on several different plots to select data points based on more than one metric simultaneously.

Using this system it is easy to inspect whether any unusual values in the plots are the result of artifacts. The suspected values are brushed, and the 3D view is then used to inspect the object, the intermediate representation and the measured values. The appropriate inspection locations are easy to find, since they are the only ones with a graphical representation of the measured values.

5.2.1 Application

We have used the system to create a concrete application for the analysis of the shape of our corals. Scans of real corals contain numerous artifacts, with. the result that some measurements do not represent what would be the correct value for that particular branch, while other measurements are actually carried out on something that is not really part of the actual coral and should be ignored altogether. Some artifacts can already be spotted and remedied during the processing that is performed prior to analysis, but due to the complexity of the structure many artifacts remain unnoticed.

The 3D visualization contains a translucent surface rendering of the coral, which can be made opaque to make inspection easier, the 3D skeleton inside the coral, and geometric representations of the measured values. The thickness metrics are rendered as colored spheres of the appropriate diameter, while the angles are represented by lines or tubes with an arc. The spacing between the branches is rendered as translucent spheres. Each metric can be turned on or off altogether if desired.

The plots can be either histograms of a single metric or scatter plots of two metrics. It is also possible to draw separate data ranges for each branch order. Some of the metrics vary depending on this order; a value might fall within a normal range for the coral as a whole, but be unusual in the actual branch order in which it was measured.

In the following sections we give an account of a number of different artifacts encountered in the measurement data, which can be located using our system. These artifacts can be present in the original coral, they can be the result of limitations in the algorithms used to process the 3D images, or they can be a side-effect of the implementation of the metrics.

5.2.2 Artifacts in the Original Data

Some of the artifacts are actual features of the coral, but are undesirable for the purpose of shape analysis.

Tube worms of various kinds are a common sighting on corals. These small worms live inside calcified tubes, usually extending from these tubes, but they withdraw into them at the first sign of danger. The bottom of these tubes needs to be attached to something, and quite often they can be found attached to coral specimens. These tubes are made out of essentially the same material as the calcified coral skeleton, thus on a CT scan they will have the same density value. This causes segmentation algorithms to regard them as part of the coral, and as such they are measured along with the actual coral. A close-up rendering of these tube worms is shown in Figure 5.4.

As the diameter of the tubes is approximately an order of magnitude smaller than that of the coral branches, these worms are an obvious source of outliers in the branch diameter measurements. Thus brushing the outliers with very small diameters will usually confirm that these measurements are tube worms. This can be seen in Figure 5.6. As the worms are not part of the coral, these ‘branches’ should be removed from the 3D skeleton data.

Holes in the coral that are located close to a measuring location affect the results of measurements, even if they have a very small diameter. This is due to the fact that the thickness of branches is measured by locating the closest point on the surface. Since there are worms or other organisms that tunnel through corals, as well as probes of the material being taken by scientists studying the composition of the coral, it is likely that a specimen will contain at least a few of such holes.

Outliers of the minimum or maximum thickness values located at the lower end of the range can indicate a hole in the coral. Only inspection of the coral surface at the corresponding locations can determine whether that is indeed the case. Some method of filling up the holes should be used, or the affected values should be excluded from the analysis.

5.2.3 Artifacts from Algorithms

Small perturbations on the surface of the coral can cause skeletonization algorithms to produce spurious order 1 (terminal) branches that do not reflect the actual topology of the coral. This is an inherent limitation of many skeletonization algorithms, and is not specific to coral data. Most, if not all, of these branches can be located by brushing a scatterplot or order 1 branches with the lengths of the branches plotted on one axis against the diameter of the branches at the end of the parent branch, as shown in Figure 5.7. All data points where the branch length is less than half of the diameter indicate that the branch does not actually extend beyond the diameter of the coral at the branching point. A close-up view from inside the coral of this phenomenon in a particularly bushy skeleton can be seen in Figure 5.5.

The remedy for this kind of artifact is to first filter the skeleton for branches that are far from the selection criterion, i.e., those that are much shorter than the parent branch diameter. The remaining branches that come close to the criterion can then be inspected manually, and then removed if this is deemed necessary. Moreover, if the skeletonization algorithm creates large amounts of such branches it would be advisable to consider using a different algorithm, or to apply more pre-processing, for example noise filtering, on the data.

5.2.4 Artifacts in the Metrics

Sometimes metrics can produce wrong results in some circumstances; therefore any odd values should always be inspected.

Minimum thickness spheres, unlike the maximum thickness spheres, are not defined at a fixed location on the skeleton graph, but some searching is needed to find a suitable measurement point. There is some tolerance in finding the location, and unexpected values can indicate that the tolerance was too large. This can especially happen if one branching point is close to the next branching point.

A useful plot for finding suspicious minimum thickness spheres is a scatter plot of the minimum thickness and the branch length; a minimum thickness close to twice the length of the branch would warrant further inspection.

Branch spacing can sometimes produce unexpected results. Depending on the geometry of the skeleton, it is possible that the smallest sphere that cuts the graph into two or more pieces only encloses the measured branch itself. Measurements in the low end of the range should be checked and ignored if necessary.

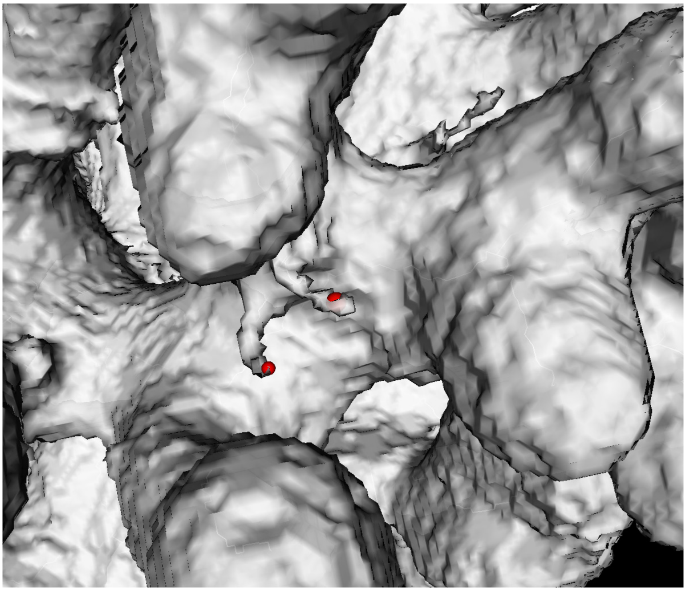


Figure 5.4: Close-up of tube worms These are clearly much thinner than the actual coral.

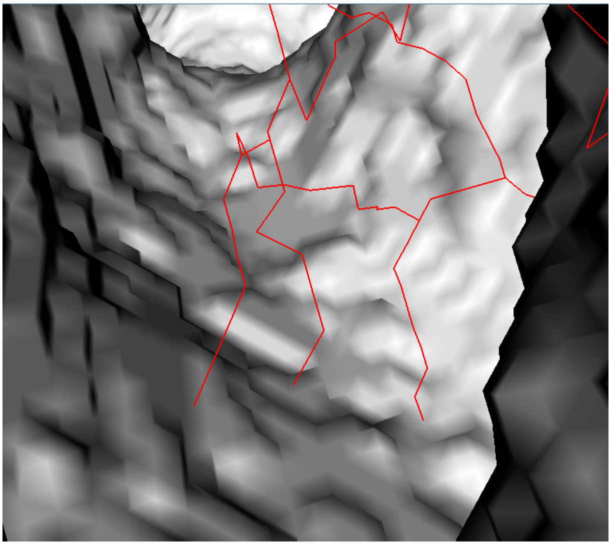


Figure 5.5: Close-up of a skeleton with spurious branches. This view is from inside of the coral branch.

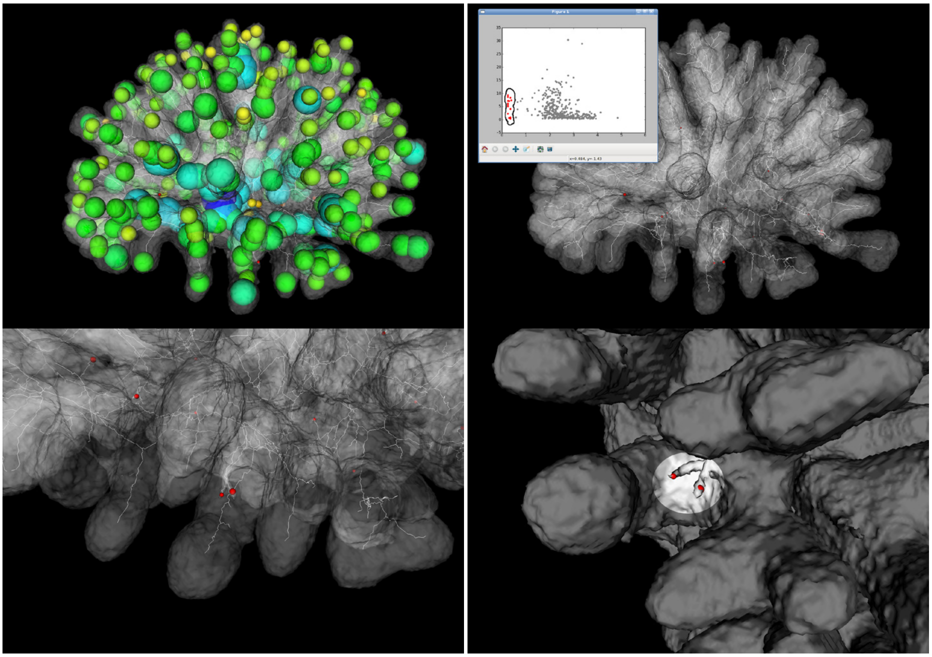


Figure 5.6: Brushing of outliers. Here it is used to locate and inspect artifacts in the linked 3D view.

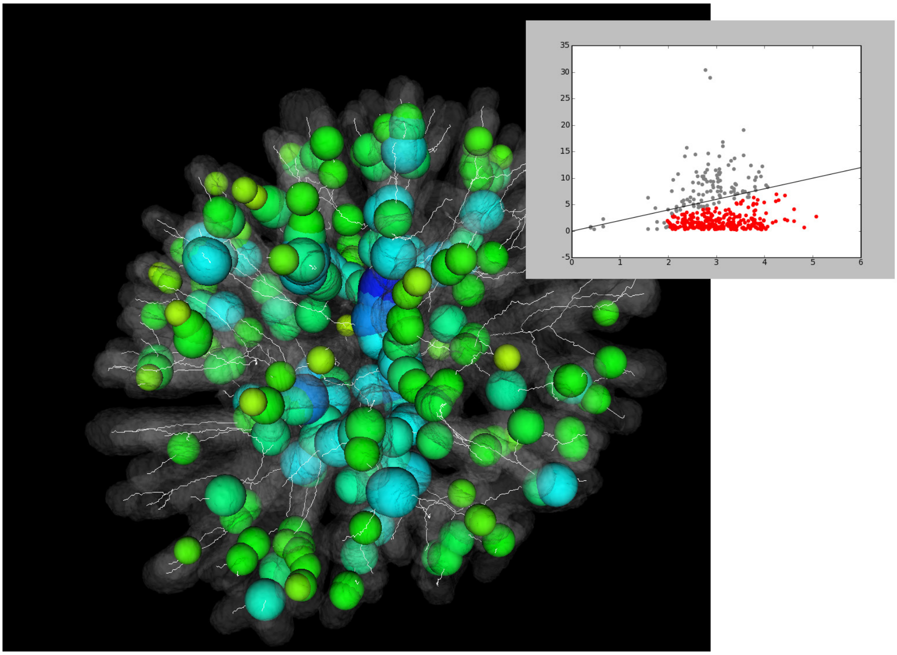


Figure 5.7: A particularly bushy skeleton. The selected measurements are those where the length of the branch is smaller than half the diameter of the maximum thickness sphere of the parent branch, which would indicate spurious branches. The diagonal line is the limit where the length equals half the diameter.

5.3 Uncertainty Visualization

Scientific visualizations usually present data as if it were absolutely correct. The uncertainty of the data, resulting from simplifications, assumptions, sampling and errors, the choice of different processing and visualization algorithms, and many other sources, is often left to the imagination of the user, in spite of a large body of research into uncertainty visualization.

A great effort is put into rendering visualizations as photorealistic and accurate as possible, but such realism leads to a false sense of certainty and absoluteness of the data being visualized. Often a level of detail is visualized that is not present in the original data.

Many methods exist to add uncertainty information to a visualization. Uncertainty can be visualized by using existing attributes of a visualization, such as opacity or color, to indicate the amount of uncertainty. Another possibility is to modify the geometry of the visualized data.

There are a number of parameters that characterize an information channel. A channel can be discrete or continuous. A discrete channel has a (possibly unlimited) number of discrete steps; a continuous channel does not. Regardless of this, there can be a limit as to how many different levels can be discerned in the channel. Another parameter is the spatial resolution of the channel. The linearity of (the perception of) the different levels is also an issue, as a linear increase in the actual value can be perceived as a larger or smaller actual change.

Some methods of uncertainty visualization, such as mapping the uncertainty to color, saturation, transparency, or other graphical attributes, make that attribute unavailable to visualize other variables. Animation of uncertain data is only suitable for interactive applications, and only makes sense when the uncertainty concerns the location of the data. Adding glyphs can create visual clutter, and blurring or geometry distortion, while a natural choice, can obscure the original geometry, and is also only natural for positional uncertainty. Using varying degrees of photorealism keeps other attributes available for other data variables to be visualized; the photorealism is then simply an additional varying attribute in the visualization.

5.3.1 Related Work

Many methods of visualizing uncertainty have been explored to date. A survey of these methods has been published by Pang et al. [62]. A more recent survey was performed by Johnson and Sanderson [38]. Recent theoretical analyses of uncertainty visualization frameworks have been made by Thomson et al. [80], and by Zuk and Carpendale [91].

Error and uncertainty are frequently visualized using existing graphical attributes, such as color or opacity. Uncertainty is also visualized by modifying the geometrical representation of the data, or through blurring of uncertain parts of a rendered scene. Error and uncertainty have been visualized using procedural annotation of the data by Cedilnik and Rheingans [14].

Point-based rendering has been used by Grigoryan and Rheingans, with random point displacement along the surface normal with a magnitude corresponding to the uncertainty of the surface [30,29].

Rheingans and Joshi have visualized positional uncertainty of molecules using a number of methods, including superimposing discrete possible locations, and rendering a likelihood volume [66]. Wittenbrink et al. have used special glyphs to convey the uncertainty of a vector field, showing directional uncertainty together with the direction and magnitude of the vectors [86]. Djurcilov et al. have added speckling and noise to volume rendered data to indicate uncertainty [23]. Strothotte et al. have used non-photorealistic sketchy rendering to express the uncertainty of virtual archaeological reconstructions, where the uncertainty increases towards the top of the reconstruction, away from the uncovered, and highly certain, foundations of the structure. The whole reconstruction is rendered using outlines, with uncertainty expressed using either increasingly sketchy or faint line styles [77].

Non-photorealistic rendering techniques are inspired by a wide variety of artistic illustration styles, such as drawing, animated cartoons, painting, and technical illustrations. In addition to artistic and entertainment uses, non-photorealistic rendering is often used to simplify the presentation of otherwise complicated objects [69,75]. Sketchy renderings are often preferred to convey the preliminary state of drafts and design concepts [57]. Some of the specific techniques used include drawing object outlines [58], hatching strokes [48,64,85], and the use of non-photorealistic lighting models [27]. Various other drawing and painting techniques have also been implemented, such as watercolor [19] and oil paintings [31], and charcoal drawings [50]. Non-photorealistic rendering styles have been implemented in image space, with the use of fixed-function graphics hardware, in programmable graphics hardware, and by combining these approaches.

Jesse and Isenberg have used hybrid photorealistic and non-photorealistic rendering styles to draw attention to specific objects or parts of objects in a scene [36]. The blending of photorealistic and non-photorealistic styles has been used by Jesse et al. as an additional dynamic graphical attribute, again to increase the saliency of parts of a rendered model [37].

5.3.2 Approach

In our approach we locally vary the rendering style of the visualization between a photorealistic style and a sketch-like artistic style, depending on the uncertainty of the underlying data. This is achieved by using special programmable vertex and pixel shaders, which are supplied with a certainty parameter at each part of the data. The data is rendered twice by the shaders, one time using the original rendering technique, and one time using a non-photorealistic sketching method. The certainty parameter is then used to combine the two renderings, with more certainty resulting in a more realistic rendering. A number of different non-photorealistic rendering styles are shown in Figure 5.8, in the vertical direction. We have chosen hatching as our non-photorealistic rendering style, because this still allows the other graphical attributes, such as opacity or color, to be used for other data variables in the same visualization. The fuzzy look of hatching is also more readily associated with an approximation, as opposed to for example cartoon-style outline drawings, which imply more exactness.

A linear mapping of the certainty of the data to photorealism is not necessarily the best option. Slightly artistic rendering may already suggest highly uncertain data, while this may not be the case. We have therefore experimented with a number of different mappings. In addition to a linear mapping, we have used a square root mapping, a quadratic mapping, and we have also used a discrete quantization of both these mappings into four steps between 0.0 and 1.0. Examples of these mappings are shown in Figure 5.8.

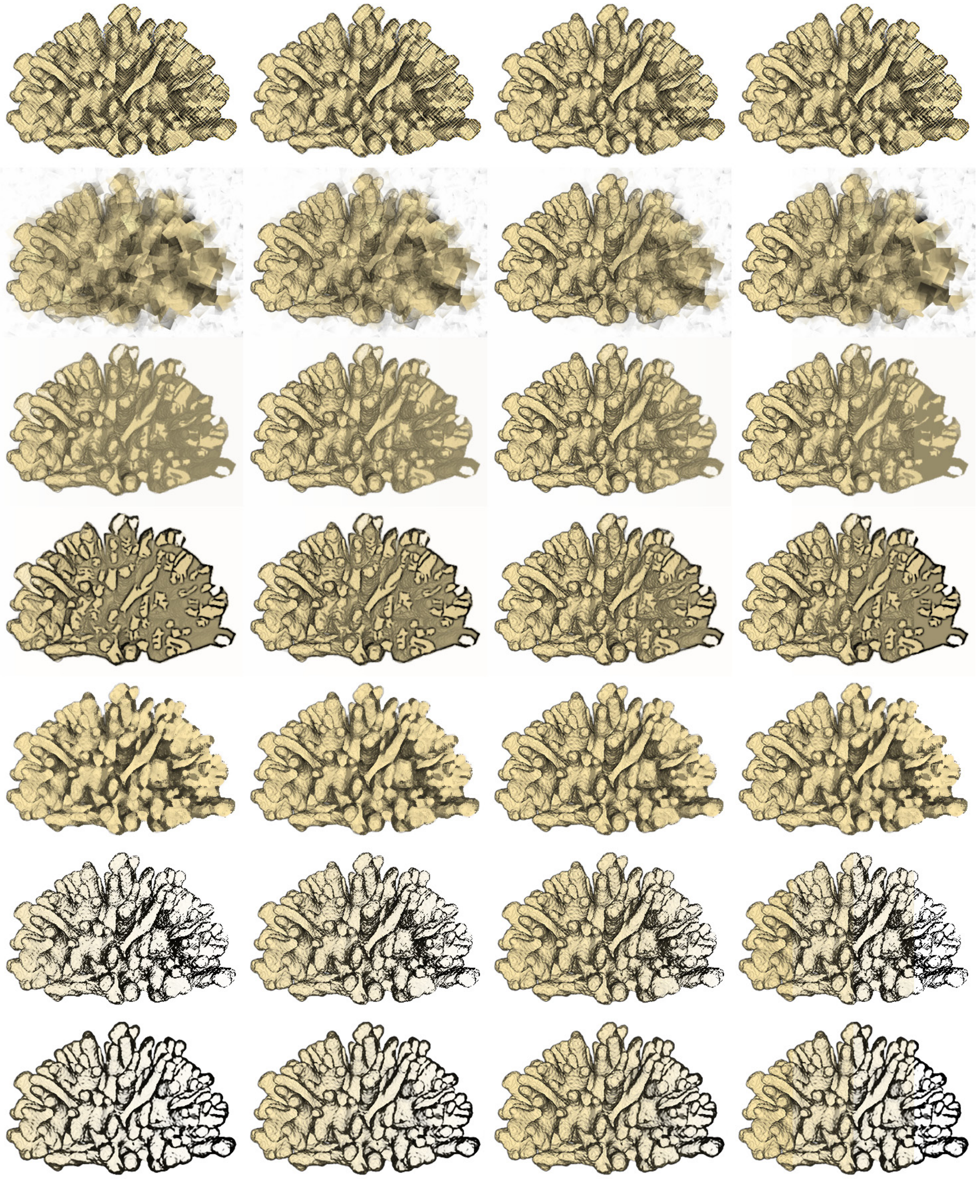


Figure 5.8: Various rendering styles. From top to bottom: crosshatch, cubism, cutout, cutout with edge silhouettes, dry brush, graphic pen and photocopy styles. From left to right: square root, linear, quadratic, and discrete mappings, with the uncertainty increasing towards the right in each image.

The range of the uncertainty is always normalized between 0 and 1, therefore a quadratic mapping will show only very high uncertainty in a non-photorealistic style, while a square root mapping will show data with lower uncertainty in a less realistic style. A discrete mapping can be useful when a more abrupt transition between different levels of uncertainty needs to be visible, especially when the change in rendering style would otherwise be too subtle to notice. The optimal mapping depends on the specific data that is visualized, as well as on how much the particular rendering style differs from the regular photorealistic rendering.

The method can be used for different types of visualizations. We have applied it to the visualization of surfaces with uncertainty, and to the visualization of line data, in the form of skeletal graphs of 3D images.

5.3.3 Surfaces

Our non-photorealistic rendering of surfaces consists of creating the impression that the surface has been drawn using a pencil or a similar tool, with the use of a hatching technique. The hatching strokes are created by using three overlapping textures containing black strokes of various length. The textures are applied to a local patch consisting of a number of polygons of the surface, and the textures are rotated to align with the direction of the local minimum principal curvature [78] at the center of each texture patch [48]. To create brighter and darker hatching tones, textures with more and less strokes are combined into a three-dimensional texture. This third dimension contains images with increasing numbers of strokes, and the value resulting from the regular light calculation is used to index the depth of the textures, to produce a brighter or darker image. To maintain a consistent stroke size across the whole scene, the textures corresponding to lower detail mipmap levels all have the same stroke width in pixels, thus the stroke width will never differ by more than a factor two [64].

Our hardware vertex program performs the normal lighting calculations, as well as a slightly contrast enhanced calculation. This second calculation is then used as the depth coordinate for the 3D textures. The pixel shader, also called a fragment program, receives the color and texture data, as well as the certainty value. After texture lookup, it performs a linear interpolation between the color value from the texture, and the color value from the normal color calculation.

5.3.4 Lines

Lines are often visualized either by creating 3D tubes or ribbons along their path, or by projecting the endpoints onto screen pixel space and drawing 2D lines between those endpoints. In case a tube or ribbon is used, the same method as for surfaces can be used. However, if the lines are visualized as 2D lines, a different approach is needed. A method is needed to draw a sketchy line instead of the 2D line, while retaining the specific properties of 2D lines, especially their constant on-screen width. Additionally, drawing 2D lines is not a photorealistic rendering method, which limits the number of possible artistic rendering methods with which it could be combined.

To create a sketchy line visualization, again stroke textures can be used. However, because of the limitations of texturing 2D lines with 2D textures, we have used special 3D ribbons instead of 2D lines. During pre-processing, each line segment is expanded into a part of a ribbon, and information about the original line is added to the ribbon. When rendering, a vertex shader orients the flat surface of these ribbons towards the screen, and also scales their width proportionally to the perspective division, in order to maintain a constant on-screen width for the line, regardless of perspective or rotations. These ribbons can then be textured with an appropriate texture, to appear as a solid line, or as sketched strokes. The texture coordinates are also transformed in the direction of the ribbon. The *s* texture coordinate is divided by the perspective division, which, together with the ribbon width correction, results in a constant on-screen size of the texture.

We have used two textures for the lines, one texture of a solid line, and one texture of a number of wavy, faint lines. The parts of the texture without any strokes are completely transparent. Again, the fragment program performs an interpolation between the straight and wavy lines, depending on the certainty value. A transition from uncertain to certain line data is shown in Figure 5.9. The two endpoints of the line may or may not be located at the same distance from the viewer; as with normal lines, this is impossible to tell without interacting with the visualization.

Figure 5.9: A line with varying uncertainty. On the left the uncertainty is high, and on the right the uncertainty is low.

5.3.5 Results

For each visualization type, we have compared the four different certainty mappings with one another, as well as with a different type of uncertainty visualization.

- **Surface Visualizations**

We have applied our method to indicate a measure of uncertainty in a visualization of one of the coral datasets. The measure used here is the magnitude of the 3D image gradient vector of the filtered CT scan, sampled along a smoothed iso-surface of the segmented coral. The segmentation of an image is more likely to be correct in locations with a high gradient magnitude, i.e., where the values of neighboring voxels exhibit a large difference. This is the case for voxels that lie along the sharply defined edge of an object, which is exactly where a correctly segmented region is expected to end. Areas with a low gradient magnitude indicate gradual change in the voxel values, thus soft or absent edges in the image. Thus, if the border of a segmented region of the image passes through such an area there is less certainty about whether the segmentation method correctly identified the boundaries of the coral.

When using the linear mapping, as shown in Figure 5.10, it becomes obvious that most of the uncertainty of the data falls somewhere in the middle to low range. The hatching is most pronounced around the tips of the branches. The quadratic mapping, shown in Figure 5.10 provides good clues as to where the most uncertain parts of the data can be located. A close-up of the visualization is shown in Figure 5.12. The combined color and hatching visualization shown in Figure 5.11 provides the most detailed uncertainty information. The uncertainty is mapped linearly to the color, with the extreme high and low values showing as red and blue respectively, and the uncertainty is also quadratically mapped to the artistic rendering style, as can be seen from the hatching in high uncertainty areas. Figure 5.13 is a direct rendering of the uncertainty value to grayscale values, with more uncertainty shown in a darker shade.

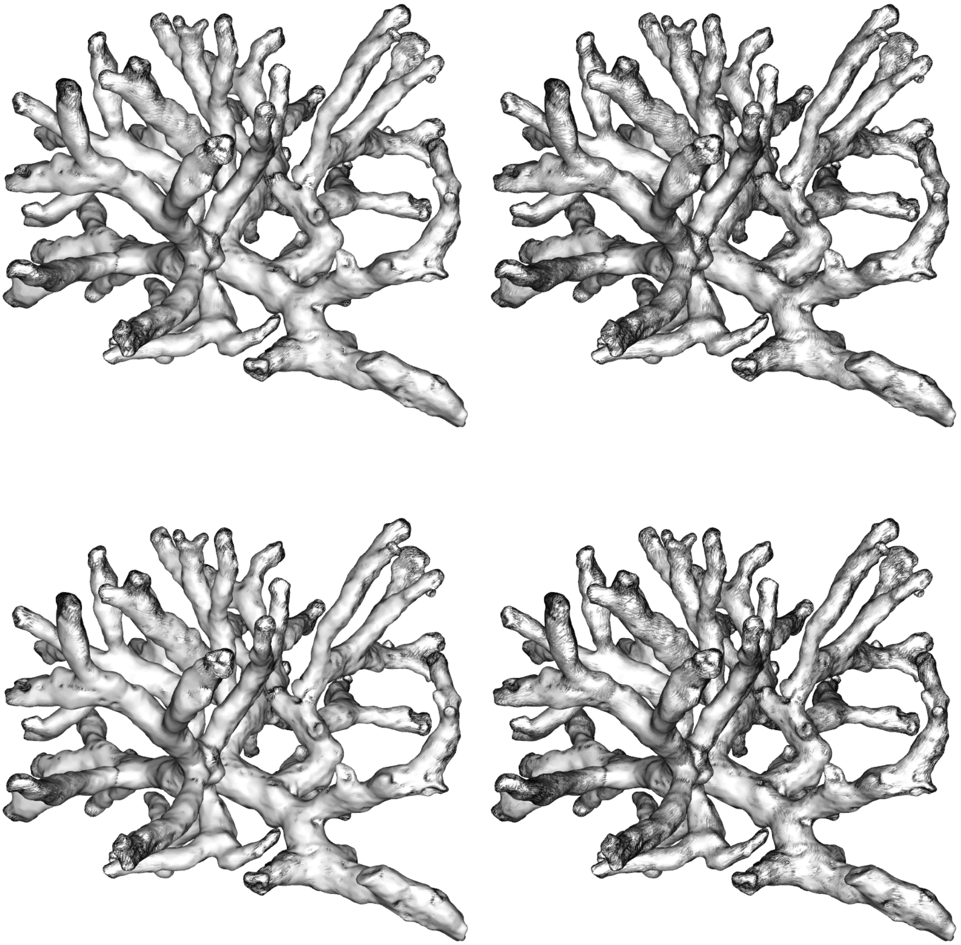


Figure 5.10: Mapping of uncertainty to photorealism. On the top are the continuous mappings, on the bottom the discrete mappings. On the left are the quadratic mappings, and the linear mappings are on the right.

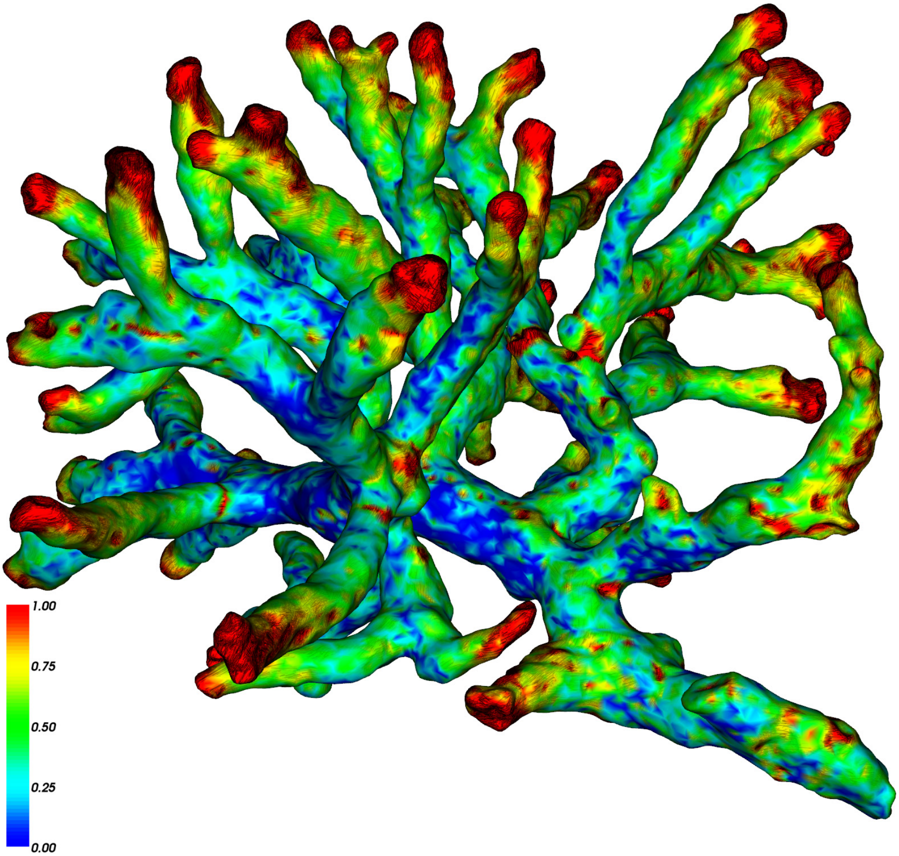


Figure 5.11: A combined visualization of uncertainty. Hatching indicates uncertainty, and is mapped quadratically. The color also represents uncertainty, but is mapped linearly from high uncertainty (red) to low uncertainty (blue), with the average values in green. This also demonstrates the use of an artistic rendering style orthogonally with color.

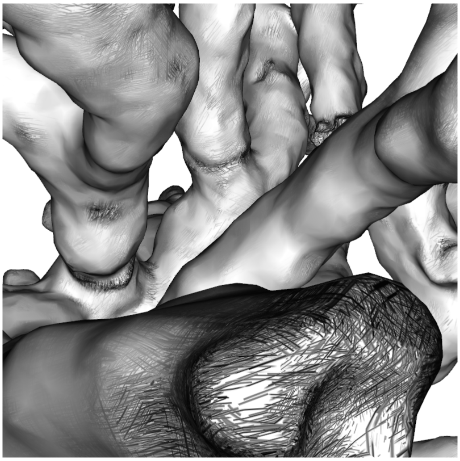


Figure 5.12: A close-up of the coral surface visualization with uncertainty. Here a continuous quadratic mapping of uncertainty to photorealism has been used.

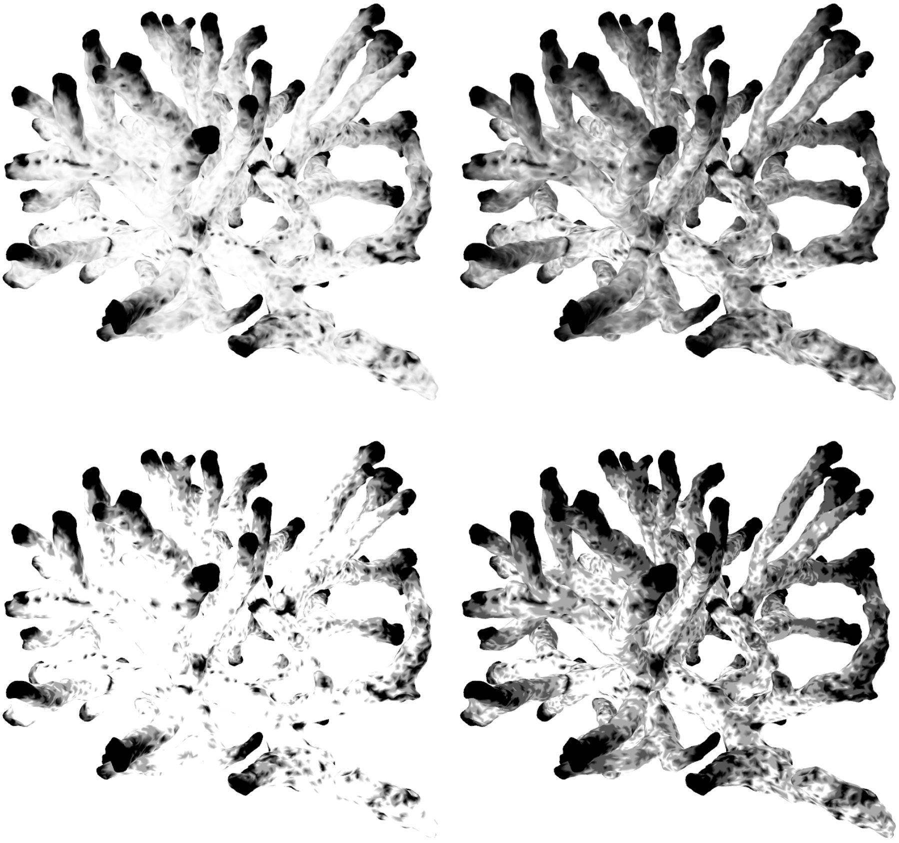


Figure 5.13: Direct rendering of uncertainty to grayscale. Darker means more uncertainty. From left to right quadratic and linear mappings, from top to bottom continuous and discrete mappings.

- **Line Visualizations**

We have applied our method to visualize the uncertainty present in the skeleton graph of one of our corals. The uncertainty in this skeletal data set stems from the fact that thinning algorithms will often yield slightly different results, depending on the order and direction in which the discrete 3D image data is processed. For example, if the object has a width of an even number of voxels before thinning, then the single-voxel centerline will fall on one of the two most central voxels. The outcome of the thinning is determinate, but which voxel is selected will depend on the direction in which the data is processed.

The uncertainty of the data has been determined by extracting the skeleton from the image after transforming it into a number of different orientations, performing the inverse transformation on the skeletal image, and assigning to each voxel the number of orientations in which that voxel is part of the skeleton. We have only used the 48 possible orientations resulting from 90-degree rotations and mirroring operations, as these can be carried out without any re-sampling. The voxels thus have a certainty value ranging from 1 to 48, with 1 indicating that only in a single orientation the voxel is part of the skeleton, while 48 indicates that the algorithm will mark that voxel as part of the skeleton regardless of the orientation of the data. These values are then sampled at the location of the skeleton in the non-transformed data, to produce the certainty of a single skeleton. The uncertainty data is normalized between 0 and 1.

A skeletal line graph (see section 4.1) is constructed by connecting these voxels, and the lines are visualized using the aforementioned method using textured, oriented, and scaled ribbons.

Because the certainty of the skeleton is measured as the fraction of orientations in which a voxel is part of the skeleton, it makes sense to use a square root mapping between uncertainty and the visible graphical attribute used to indicate this uncertainty. Near the bottom of the range, it makes a large difference whether a voxel is in the skeleton for 3 or 4 orientations; near the top, the difference between 47 and 48 orientations is almost negligible. However, a linear mapping would indicate a similar difference in certainty for both.

An overview of the whole skeletal graph, as well as a close-up of the coral skeleton can be seen in Figure 5.14. Both use the discrete mapping of the square root of the uncertainty. The more pronounced black lines have a higher certainty than the fainter gray parts.

Figure 5.15 is an image of the surface technique applied to a tube representation of the same skeleton graph. Here the square root of the uncertainty is mapped to the hatching rendering style.

5.3.6 Discussion

For the coral surface, the difference between the continuous and discrete versions of both mappings is rather subtle in the final images, despite the clear difference visible in Figure 5.13. This indicates that the number of discernible levels of photorealism in such blended images is not very high. The quadratic mapping is much better than the linear for locating special attention areas with high uncertainty, although if that were the main purpose of the visualization, then using a color-visualization would provide much more detailed and salient information. However, if uncertainty is not the main concern, the hybrid visualization leaves the color attribute available for other variables to be displayed.

Most of the uncertainty can be found at the tips of the branches. This corresponds to areas of low density in the coral. The combination of color and hybrid rendering styles provides the best distinction between low and high certainty, but this is expected when two attributes are combined to visualize the same quantity. The combination of color and photorealism is particularly valuable to express with non-photorealism the uncertainty of a data variable that is in turn mapped to color and sampled along a geometry.

The linear mapping for the skeleton uncertainty, as expected, suggests a greater uncertainty than is actually present. The quadratic mapping is corresponding better to the actual uncertainty of the data. The difference between the quantized and continuous mappings is not very pronounced, but more visible than with surfaces.

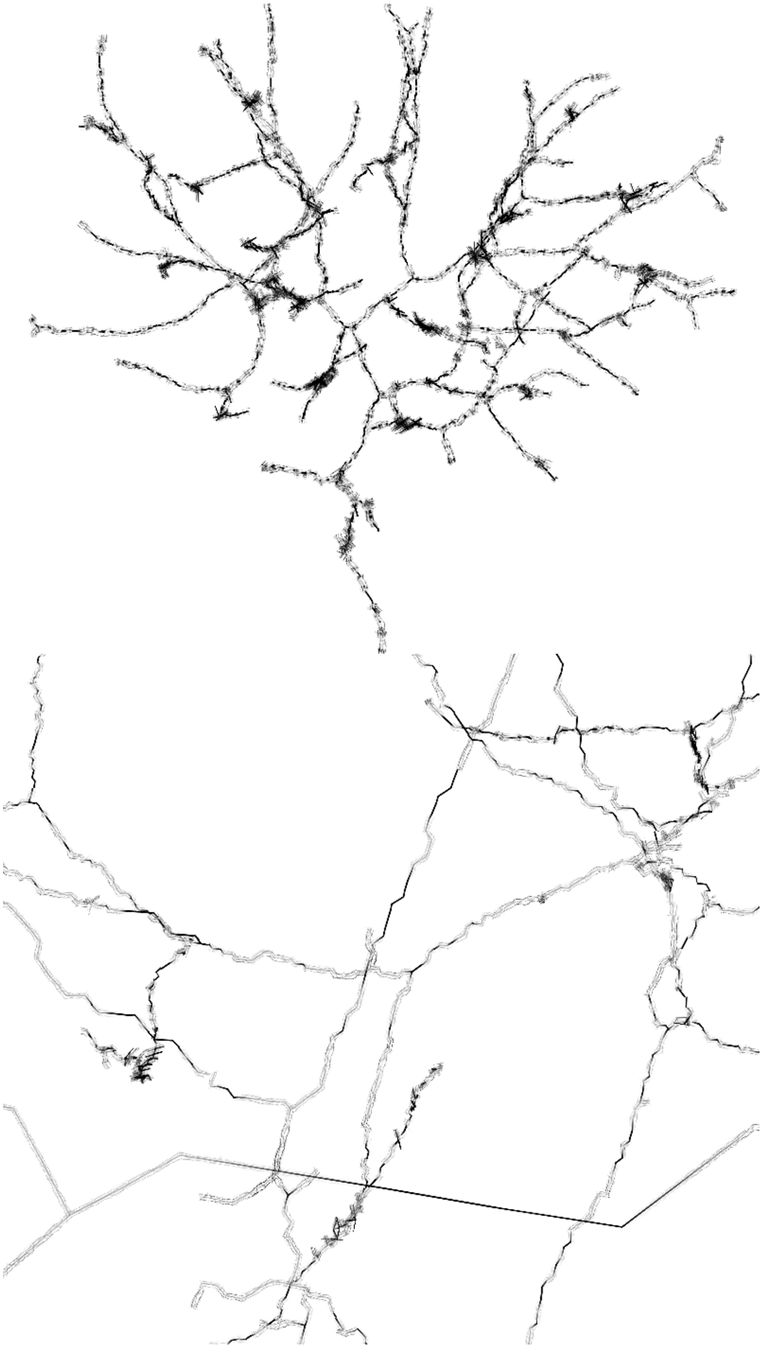


Figure 5.14: Skeleton graph visualization with uncertainty. The whole graph (top) and a close-up (bottom) are shown. The well-defined black lines denote high certainty.

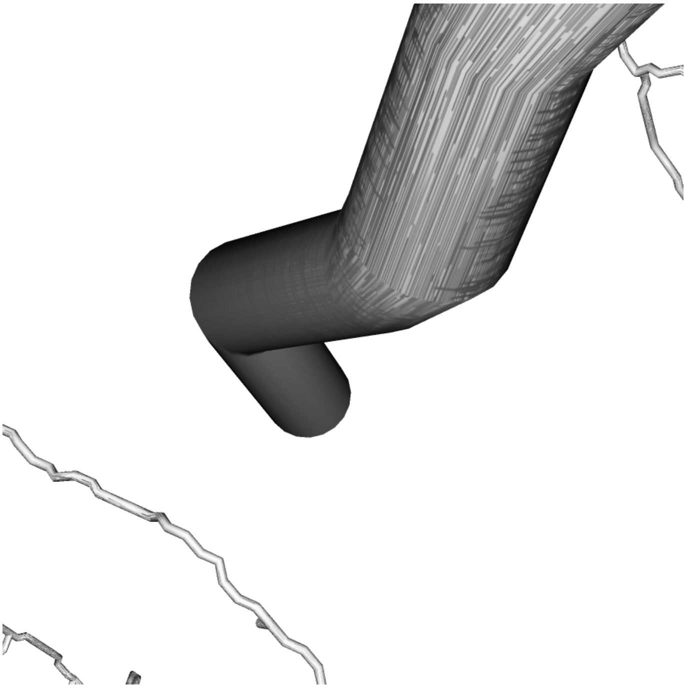


Figure 5.15: Skeleton data set with uncertainty visualized using the tube approach. The square root of the uncertainty determines the level of realism.

The coral skeleton dataset consists of long lines spaced far apart, with densely spaced highly fluctuating certainty values. Using the tube method of line visualization has the disadvantage that the tubes become very thin when the data is viewed as a whole, and thus the presence or absence of hatching is hard to determine, making the method less suitable for a global overview of the uncertainty of the skeleton. The line method is also not very successful at giving an overview of the uncertainty of the skeleton, as many points are compressed together on a small part of the screen. A more hierarchical, level-of-detail based method is needed for this purpose.

In general, while it is intuitive that the sketchy parts of the data are uncertain, the number of discernible steps on the scale between normal and artistic rendering is rather small. This makes hybrid non-photorealistic rendering less suitable for an analysis of the data where the uncertainty is the primary concern. It is, however, very well-suited as an intuitive indication of more and less certain parts in the data, especially when other graphical attributes, such as color, are already being used for the visualization of a different data variable.

5.4 Conclusion

In this chapter, the use of a number of visualizations was discussed. These visualizations are used to display a static summary of the data. They are also used to interactively explore patterns in the data, and to modify the data.

Interactive editing is a necessary tool that is used throughout the measuring process. It is required for some actions which can only be performed manually, while it also serves to control and verify automated processing.

Interactive data exploration with linked 2D and 3D visualizations is a powerful tool to gain better understanding of the coral data. It enables the user to find links between numerical and spatial features, and to quickly determine the origin of statistical outliers. On the other hand it is also used to interactively perform statistical analysis of specific regions of the coral.

Uncertainty is always present in data, but it is not always explicitly shown in visualizations. One method to display the uncertainty in 3D visualizations is to reduce the photorealism of the rendering. This is a natural choice, as realism is intuitively linked with certainty. The use of sketch-like rendering methods for surface and line visualizations was analyzed. While the rendering method does convey uncertainty, the limited number of discernible levels makes it mostly suitable to indicate areas with higher uncertainty, rather than being used to visualize the exact degree of uncertainty.

Chapter 6 Tangible Interaction

This chapter deals with tangible interaction. Different parts of a user interface can be tangible. On one hand tangible interaction can provide a physical extension of elements of the user interface, such as 3D widgets, making it easier to manipulate them. Conversely, the data itself can be made tangible, by using 3D printed reproductions to create a natural user interface, manipulating the data in the same way everyday objects are used.

Tangible interfaces provide a more efficient method of interaction than regular 3D input devices. This is demonstrated in the literature, and we have confirmed this in a user experiment. In addition, an input prop that corresponds to the object being manipulated can sometimes offer additional benefits over a generic device, especially for complex objects. We hold a well-founded belief that this is the case for corals, where we use a printed replica as a tangible prop.

6.1 Introduction

Input devices have been traditionally generic pointing devices, ranging from the two degrees-of-freedom mouse through six degrees-of-freedom devices (e.g., wands, gloves, etc) and to haptic devices. Buttons are placed on the devices to trigger actions. In all cases, however, these devices are used to control a single point location in the 3D interaction space.

Interaction with three-dimensional data often involves 3D user interface widgets, proposed in [17]. A widget is a combination of geometry with certain behavior that is used to control the visualized data, or to query information about that data. The widget is placed in the 3D scene and can then be manipulated. Examples of 3D widgets are widgets for probing data values inside a data set, placement of seed points for creating streamlines in a vector field, placement and orientation of slicing planes, drawing and manipulation of selection contours, manipulation of bounding boxes for extracting part of a data set, or scaling, orienting and placing a geometric structure. The widgets typically have a number of handles that can be manipulated to obtain the desired effect.

An important feature of interaction with 3D widgets is that restrictions can be placed on the way in which the handles of the widget, or even the whole widget itself, can be manipulated. Examples of this are a widget where a handle can only be moved across the surface of a sphere, or a widget to select a location within a bounding box. Such restrictions actually make the widgets easier to use, as the input is now confined to what makes sense in the application.

It is our belief that providing users with only pointing devices results in non-intuitive and often difficult to use user interfaces. In our view, the main reason for this is that the physical devices serve only as a proxy for a virtual object, and the user has to rely on the generated visual or haptic feedback to interpret the effect of her actions. The actual interaction occurs in a remotely controlled and immaterial virtual world, distinct from the user's familiar environment. Notwithstanding the use of expensive haptic devices, attempting to pick locations in thin air guided by nothing more than imperfect visual feedback is an inefficient method of interaction. While this may be acceptable for interaction with objects that have relatively simple geometries, for the complex three-dimensional branching shapes of coral colonies a better interface is required.

Tangible user interfaces provide real, physical props that can be manipulated by the user, and that have a virtual counterpart. These input props are identical, or at least similar in shape to their virtual counterparts, and both can be manipulated in the same way. Thus the user interacts by performing exactly those actions on the tangible prop that he would like to perform on the virtual representation. In a sense, the user holds the virtual object in his hands and can directly interact with it, instead of only manipulating it indirectly. This direct manipulation of physical objects provides a very natural and easy to use interface, since it taps into the same skills that the user needs to sense and manipulate his physical environment in daily life.

In this chapter, we explore how tangible input props can be used to provide a better interface for interaction with coral data in virtual and augmented reality. Our approach uses rapid prototyping technology to print three-dimensional physical representations of the coral data, which are subsequently imported into the environment and used as tangible devices for interaction. The approach is iterative; so that more complex props can be used as more insight into the underlying simulation is gained.

A very important feature of the approach is that actions on the tangible coral props can be initiated without the use of buttons, instead relying on detection of contact between the props and a stylus.

This tangible prop can be used as a more natural interface for two purposes. It can be used to interactively perform measurements, by probing locations of interest on the prop. It can also be used for the interactive editing tasks described in section 5.1.

The work in this chapter makes use of the 3D input extension to the Visualization Toolkit (VTK [71]), described in Appendix A.

The remainder of this chapter is structured as follows. The second section discusses related work in the area of tangible interaction. The third section describes the concept of printed tangible props. The fourth section presents the advantage of a tangible interface over conventional 6-DOF input devices in a user study. The fifth section discusses how tangible props can be used and which factors affect their effectiveness. Section six discusses tangible props in combination with augmented reality. An application with printed tangible coral props is presented in section seven.

6.2 Related Work

Many scientific visualization and virtual reality systems have been developed that use a variety of tangible props as input devices. These systems use either very generic abstract props, or detailed props that are limited to a specific application or type of data. In these systems, physical props are mostly used for their resemblance to the virtual model and familiarity when touching the prop, while interaction is generally done with virtual devices and props. Haptic feedback comes from specialized devices like the Phantom, to which sometimes props are attached. Often mixed props are used in such a setup, where only the part that is handled by the user is an actual prop, while the rest of the prop exists only in the virtual world.

Hinckley et al. [32] tracked passive props to create a natural interface for a neurosurgical visualization application. To create a linear trajectory into the patient's brain a stylus was used, which controlled a ray, while the brain model was controlled with a rubber sphere. The trajectory was determined by the intersection of the ray with the brain model. A third prop consisting of a tracked piece of plexiglass was used to control a cutting plane to cut away part of the brain. Replacing the sphere with the head of a doll to provide a more realistic prop gave users false expectations, as the head prop was not exactly matched with the brain data. Placing the trajectory or cutting plane at a very specific place on the head prop, for example the eyes, would not result in the virtual tool being placed in the corresponding region of the brain model. This indicates that props should either be very abstract or an exact match of the data.

Couture et al. [18] used tangible props as input devices with a tabletop projection system for a geoscience data analysis application. One or more tangible props of various kinds were moved around a flat projection to select cutting planes through a three-dimensional volumetric model compiled from seismic data. More than one prop could be used simultaneously. A tangible box with physical buttons was used to initiate various actions. Each of these props corresponded to a virtual handle of a cutting plane, and as such the prop only served as a handle for the actual tool. The actual shape of the props or any contact made between the props had no particular meaning. They demonstrated that a specialized tangible interface resulted in better performance than using generic mouse and keyboard input.

Fitzmaurice et al. [25] controlled virtual objects through tangible ('graspable') physical handles called 'bricks'. Ishii and Ullmer [35] took this concept further with Tangible Bits, employing a more direct correspondence between physical and virtual objects in a context where almost every part of the environment could be used for interaction. However, the actual props used in this concept serve as physical handles for virtual tools, or are physical user interface elements. The data with which the interaction takes place only exists in the digital domain, and has no matching physical prop. It was noted that new users could very quickly learn to use this interface for advanced tasks.

Gillet et al. [26] have used three-dimensional printed props of molecules in an Augmented Reality application. The printed props included attachment surfaces for ARToolKit markers that provided an implicit calibration of the props. Multiple ARToolKit markers were used for each prop, also to improve accuracy but mainly to prevent loss of tracking due to marker occlusion. The props were used to position and orient the virtual models and were augmented with various kinds of information and visualizations, but no other input devices were used in conjunction with the props. One of the noted benefits was that the tactile feedback of the model made it easier to explore and memorize the structure of the molecules.

Ortega and Coquillart [59] created an immersive industrial automotive application with a mixed tangible prop, attached to a string-based haptic system. The system was used to simulate various industrial assembly tasks. The handle is a physical tangible prop, while the rest of the mixed prop existed only in the virtual world. The mixed prop is explicitly used to avoid calibration errors from using a full real prop. The main focus is on the virtual part of the prop, so any mismatch between the real prop and the virtual model is of little consequence. The prop serves only as a physical handle for a virtual tool, and no physical form is used for the data, other than the haptic feedback. The authors noted that users could perform actions using the same gestures as during actual assembly work, showing the effectiveness of the interface.

Kok and van Liere [46] controlled 2D widgets with 3D tangible props. They explored the use of passive haptic feedback as a means to detect that an action can be performed on a widget. The widgets were placed on a virtual cube, which was co-located with a tracked cube prop, and a tracked stylus was used to select the widget, but to actually start the interaction a button had to be pressed, thus the tactile feedback from the surface of the cube only served to indicate that the stylus could interact with one of the widgets on the cube. The props only served as physical and mixed mode user interface elements. Both co-location and passive haptic feedback improved user performance.

Ware and Rose [84] performed a number of experiments that involved rotating virtual objects using matching and non-matching props. One significant result was that users performed rotations faster when the visual feedback came from the physical object than when using virtual imagery. This was hypothesized to be

All these systems use tangible props as input devices. However, the props serve either as abstract physical handles for manipulating objects, with no resemblance to these objects, or they are representations or physical parts of the virtual tools used in these systems, and while their shape might correspond directly to the tool they represent, the data on which they are used has no physical representation.

6.3 Tangible Prop Concept

Tangible props add the possibility to use one’s sense of touch for the purpose of interaction with data. A physical representation of an object held in the hand gives a very natural method of interaction, since people are accustomed to handling physical objects in their daily lives.

A basic method to create a tangible prop is to create a virtual counterpart of an existing object and to apply tracking to this object. Actions such as moving and rotating can then be performed directly on the objects, instead of using a generic physical pointing device to manipulate virtual objects. The objects in such a scenario are commonly associated with specific tools or controls in the application. More often than not they are simple objects that mainly act as a physical handle for much more complex virtual objects, and as such their actual shape is less relevant, with a loose correspondence between the physical and virtual object being sufficient.

A more advanced method is where a pointing device or any other tracked object is used on a tangible input prop. While the abstract pointing device is now reintroduced as an input device, it no longer serves to indicate locations in empty space, but it is instead used to indicate specific spots on a real, physical object: the tangible prop. The shape of a tangible prop then becomes much more important, as the tactile feedback is now used to quickly and accurately find locations on the surface. This method lends itself well to create tangible controls for 3D widgets, as the limitations imposed by the widget can now correspond to physical, tangible surfaces.

Rapid prototyping technology provides the possibility to create any physical object from a virtual representation. This printed object can then be used in conjunction with a tracking system to serve as a tangible input prop. The important difference here is that instead of creating a virtual representation that more or less matches an existing object, an exact physical replica of a virtual object is created. This makes it possible to use physical representations of complex data as tangible props; thus the tangible prop is no longer a handle or a tool, but it is the actual data itself.

Some types of applications are more suited for use with printed tangible props than others. Especially applications that revolve around creating, analyzing or simulating structures, models or other three-dimensional data sets are suited for this approach. As an example we consider the development of a virtual 3D model.

6.3.1 Iterative Approach

The process of development is inherently iterative. Initially a rough basic version of the model is created. Simulations or other calculations can then be carried out using this basic geometry, or the model itself is scrutinized. The results are then analyzed and the model is adapted where necessary. This refinement process is repeated until the final model is deemed satisfactory. This process is depicted in Figure 6.1.

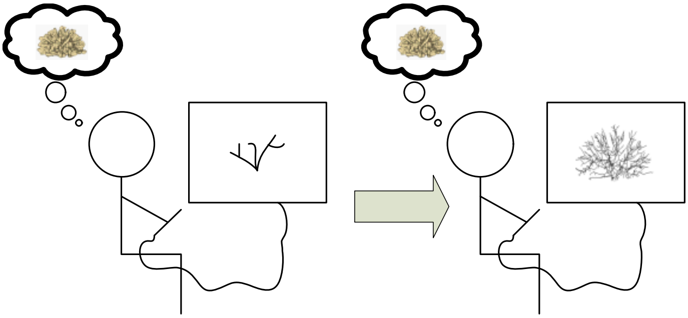


Figure 6.1: The basic model refinement process. A crude model is created and is iteratively refined.

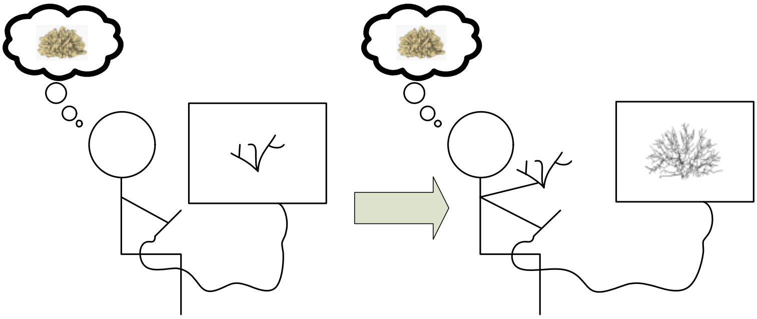


Figure 6.2: Refinement process with tangible props. A crude model is created, and a tangible prop is made from this model. This prop then serves as an input device to create a more refined model. These steps can be repeated for several iterations.

6.3.2 Tangible Prop

A natural additional step in the process is to create a real-world version of the model. A physical representation of the model can give better insights into the properties of the model. The representation can be created through various 3D printing methods, which can automatically produce a solid real-world object from the model. This technology is becoming increasingly affordable, and it is extremely likely that in a number of years 3D printers will be as ubiquitous as regular printers are today, and printed three-dimensional objects will become disposable single-use items.

If the position and orientation of the printed model are tracked and related to the original virtual model then the print can be used as a tangible input prop that corresponds exactly to the virtual model. It then becomes possible to use the sense of touch to explore the virtual model. This is depicted in Figure 6.2. The visualization system can show additional visualizations and information that is not visible on the physical prop, or show a magnified version of the model to examine small details.

The prop thus serves both as output and as input. It is the output of the last modeling iteration, a view of what the model looks like, and it can be annotated in an augmented reality system. It is also used as an input device to interact with the model.

Usually interaction is performed on virtual objects, and at most a prop serves as a proxy for a virtual input object. However, by using tangible input props the interaction actually takes place in the real world, and using a complex computer system becomes the same as using any physical everyday object. The computer system only responds to the actions that the user performs in the real world, and it is no longer the primary focus of those actions.

6.3.3 Interaction with Tangible Props

In the physical world we interact with objects by touching them. In the virtual world we usually need to press a physical button on an input device to make our intention clear to the system. The button typically has no direct relationship with the intended action.

Tangible props reintroduce a direct relationship between the virtual interaction and the physical action necessary to perform the interaction. To rotate a virtual knob a physical knob is rotated, to move a virtual item a corresponding physical item is moved.

Taking this further, touching one physical item with another, i.e., touching an object with a tool, constitutes interaction. In the physical world this kind of interaction is commonplace: we write by dragging a pen over paper, we cut bread by slicing it with a knife. We know this ‘interaction’ is occurring not only because we can see the result, but also because we feel it.

Virtual interaction in the same manner is more difficult to implement than simply relying on buttons. It is necessary to determine the differences between the physical prop and the virtual model, and to take into consideration the inherent uncertainty.

6.3.4 Haptics

Haptics refers to the use of the sense of touch in user interfaces. In the context of input devices an important distinction is made between passive and active haptics.

With active haptics, the tangible feedback is controlled by the computer and generated by an actuated input device that can exert forces in three dimensions. Depending on the actions of the user and on various calculations, the input device will provide resistance to the motions of the user, or even move the device in a different direction. Using a virtual model and the current position and orientation the force has to be recalculated hundreds or thousands times per second to provide convincing feedback.

Using an active haptic input device it is possible to simulate tangible interaction with stiff solid objects, elastic objects, viscous fluids, force fields, etc. Conceptually any interaction performed with such a device takes place in the virtual world, and the device serves as both input and output.

With passive haptics, feedback comes from the physical shape of a manipulated object. This feedback cannot be controlled by the computer, but it also does not require any calculations to determine the force, thus the feedback is 'free' and instantaneous.

Conceptually the interaction takes place in the physical world, and the computer must use collision detection on the tracked objects to determine how the interaction is performed.

6.4 Motivation

We have conducted a user study to determine the advantage of using a tangible input device. We use the tangible device to restrict the number of degrees of freedom of an input device when manipulating certain 3D widgets in a 3D environment.

While widgets are interaction techniques for manipulating data, the widgets themselves are manipulated using an input controller. These controllers vary in the number of degrees of freedom (DOF) they have. For example a desktop mouse has two degrees of freedom, while a sensor that reports position and orientation in space has six degrees of freedom, thus offering a very natural and direct way of interaction with three-dimensional objects. However, 3D widgets often place restrictions on the way in which they can be manipulated, and often not all degrees of freedom are necessary to use such a widget, while sometimes the additional freedom of movement makes widget manipulation more difficult.

Tangible 3D input controllers restrict the number of degrees of freedom, and thus can provide a method of interaction that corresponds better to the way in which a widget allows itself to be manipulated. This can especially be the case for manipulations that are in essence restricted to two dimensions, such as placing points on a plane or drawing on a surface. In addition the use of tangible controllers can give the user a better perception of where the manipulation is taking place than manipulation using an unrestricted 6-DOF controller.

We compared the use of four different input methods both for manipulating a widget for drawing a contour on a 3D surface, and for controlling the camera viewpoint. The first method uses only the mouse to control both the widget and the camera. The second method uses two 6-DOF devices for a total of 12 degrees of freedom, six for camera manipulation and six for unrestricted use of the widget. The third method uses a 6-DOF cube device as a tangible interaction surface to restrict a 6-DOF pen device to just two degrees of freedom, for a total of eight degrees. A Magic Lens [7,12,82] style projection on the tangible surface allows precise manipulation of the widget with two degrees of freedom, while the camera can still be freely manipulated. The last method retains the 6-DOF camera control, while the mouse is used for manipulating the widget. We show that in a desktop environment a mouse is the best input method for this widget, while for a virtual environment lacking a mouse the tangible controller has a significant advantage over unrestricted 6-DOF input.

6.4.1 Evaluation

We have conducted a simple user experiment on a desktop VR setup. The subject is presented with a surface representation of a part of a coral, to which a sphere has been attached. This object is shown in Figure 6.3. The task is to ‘remove’ the sphere by drawing

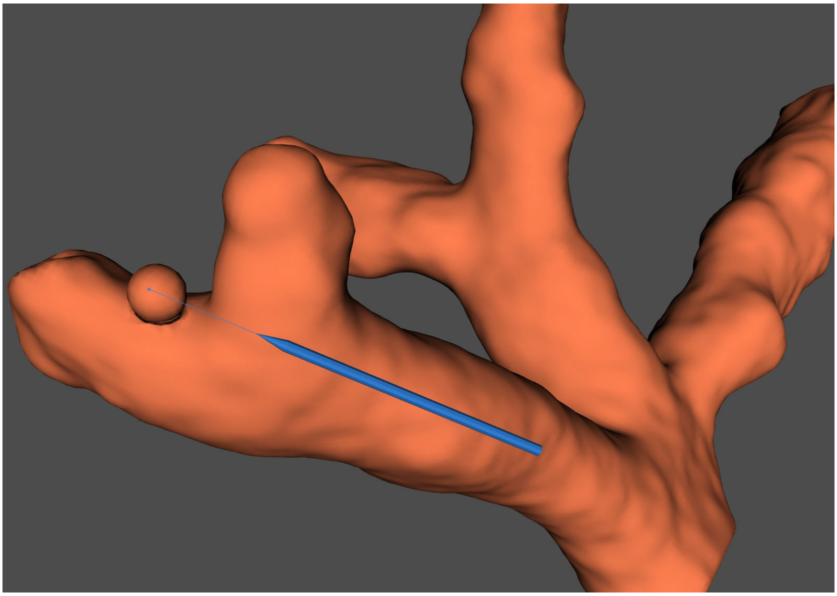


Figure 6.3: The test object with a sphere attached to it. The blue object is the virtual representation of the 6-DOF pen input device.

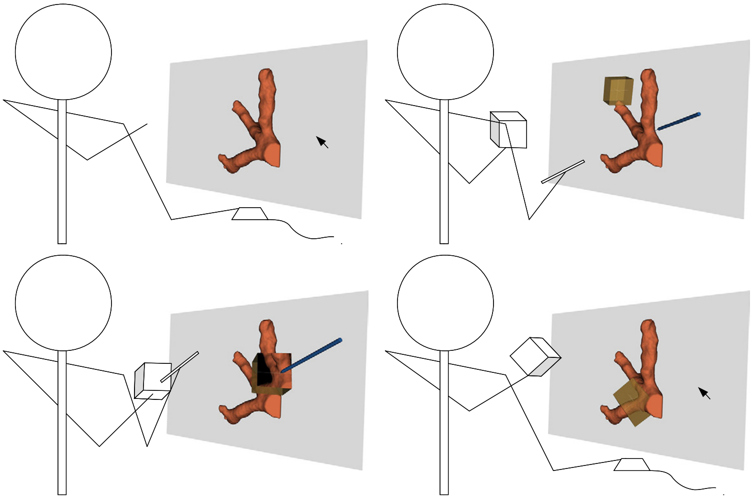


Figure 6.4: Diagrams of the four methods that were compared. Top row, from left to right: Method 1: 2-DOF mouse, Method 2: 12-DOF input. Bottom row, from left to right: Method 3: 8-DOF tangible input, Method 4: 8-DOF input with mouse.

a contour on the object, along a pre-computed ‘optimal’ line, which is shown on the surface of the object. The subject can draw a contour by marking points on the surface of the object, using various input methods to determine the location of these points. The contour widget automatically pins the points to the surface of the object, and interpolates a contour between them. Once a point is placed it cannot be modified or removed, but it is possible to select the first point, which is therefore shown in a different color. Each trial ends when the first point is selected again, thus closing the contour. The widget also includes functionality for modifying or removing already placed points, but this has been disabled for our study. As a small ‘reward’ for the subject, after closing a contour the sphere is removed from the object along the contour, and the hole in the surface is tessellated. This task is inspired by the need of coral researchers to interactively remove unwanted parts of a CT scan of marine corals before automated analysis or simulation.

Four different input methods are compared. The methods not only differ in how contour points are placed but also in the way in which the viewpoint can be manipulated. Each trial starts from the same distant viewpoint showing the whole object, with the sphere not necessarily being visible. The test subject then has to orient and zoom the view before the removal task can be started. It is also necessary to adjust the viewpoint during the task, because the ‘optimal’ contour is always partially occluded by the object.

The following input methods are compared:

Method 1 consists of selecting the location of a point on the rendered surface using 2D point and click input with the mouse. The camera is also controlled with the mouse. The orientation of the viewpoint can be rotated around a focal point by dragging the mouse cursor over the background of the 3D scene while pressing the left mouse button. Pressing the middle button and dragging moves the whole scene parallel to the viewing plane. Pressing the ‘f’ key on the keyboard picks the location of the mouse cursor as the new location for the focal point and causes the camera to move closer to this point. Rotating the mouse wheel moves the camera closer to and farther from the focal point. No 6-DOF input is used in this method. A diagram of this method can be seen in Figure 6.4.

Method 2 involves selecting the location of a point by using the 3D pen. A line protrudes from the tip of the pen, and pressing the right pedal places a contour point at the intersection between this line and the surface. The line is terminated with a thicker dot in order to make it easier to determine whether the line intersects with the surface or not: if the dot is not visible then the two intersect. The camera is controlled with the 3D cube held in the other hand, employing the scene in hand metaphor. Pressing the left pedal causes the whole scene to move along with the cube and to rotate around the center of the cube. Pressing the middle pedal and moving the pen away from the screen causes the camera to zoom in towards the center of the cube. A diagram of this method can be seen in Figure 6.4.

Method 3 uses a tangible device. In this method a Magic Lens style projection of the scene is rendered onto one of the sides of the virtual representation of the 3D input cube.

The location of a contour point is selected by touching the cube with the pen on the appropriate side of the cube at the location where the point is to be placed and pressing the right pedal. A white cross-hair cursor on the projection facilitates selecting the correct location. A diagram of this method can be seen in Figure 6.4.

The camera is controlled in exactly the same manner as in method 2 with the cube controlling the orientation and position when the left pedal is pressed, and the pen controlling the zoom towards the cube when the middle pedal is pressed. Note that while the left pedal is pressed the projection on the cube will remain unchanged when the cube moves, which allows for greater precision when selecting a contour point location on the projection. This fact is explicitly explained to each test subject.

Method 4 combines the mouse and 3D input. In this method the location of contour points is selected with the mouse like in method 1, but the camera is controlled with the 3D cube like in methods 2 and 3. The zoom function is controlled with the mouse wheel, with the difference that the camera zooms in on the center of the cube instead of the focal point. A diagram of this method can be seen in Figure 6.4.

6.4.2 Measurements

We measure the total trial time, the time needed to draw the contour and the accuracy with respect to the reference contour. The trial time is the time elapsed from the moment the test subject is presented with the view of the scene until the moment the subject closes the contour. The contour drawing time is the time between the moment when the first point is placed and the moment when the contour is closed by selecting the first point again. The accuracy is defined as the root mean square distance between the contour points selected by the test subject and the reference contour shown on the surface of the test object during the trial.

The different methods are tested in the order in which they are described. Each user is first instructed how the input method works and is given time to practice with the input method until he or she gains sufficient skill in placing contour points and modifying the viewpoint. The subject is then asked to use the method to carry out the removal task as quickly and as accurately as possible, repeating it a number of times with the sphere at different locations. The locations and the order of the locations of these spheres are previously generated randomly but are always the same for all methods and all subjects.

After finishing the test with the last method, the subject is asked to order the methods from the easiest to use to the hardest to use, and is asked which of the two methods not involving a mouse they would prefer to use in an environment where there is no mouse available.

6.4.3 Tangible Magic Lens

The Magic Lens shows a parallel projection of the scene visible in the main view using a camera position and orientation that is controlled by the 6-DOF cube, and that is constantly updated as the cube moves. This projection is rendered to an off-screen window, which is then texture-mapped to the side of the virtual representation of the 6-DOF cube in the main view. The cube thus works like the LCD viewfinder on a digital camera. The scale of the parallel projection is chosen to exactly match the size of the virtual representation of the cube, and the direction of projection is opposite to the normal of the side of the cube, so the lens will show what is directly behind the cube. This is shown with the white dashed lines in Figure 6.5. The front clipping plane of the camera coincides with the side of the cube, thus the lens never shows any part of the scene that is located in front of the cube. This makes it possible to use the lens to inspect some difficult to reach occluded areas of the scene without changing the viewpoint of the main view.

The lens also acts like a magnifying glass. Because of the parallel projection the view in the lens will have the same size regardless of the distance between the cube and the object, while the perspective of the main view will enlarge the cube, and thus the projection, when the input device is moved away from the screen. This effect can also be seen in Figure 6.5. The lens can additionally be used as a mirror, since the virtual representation of the cube is largely transparent and the back of the projection is visible through the cube. This makes it possible to perform interaction on the rear side of an object without adjusting the main view.

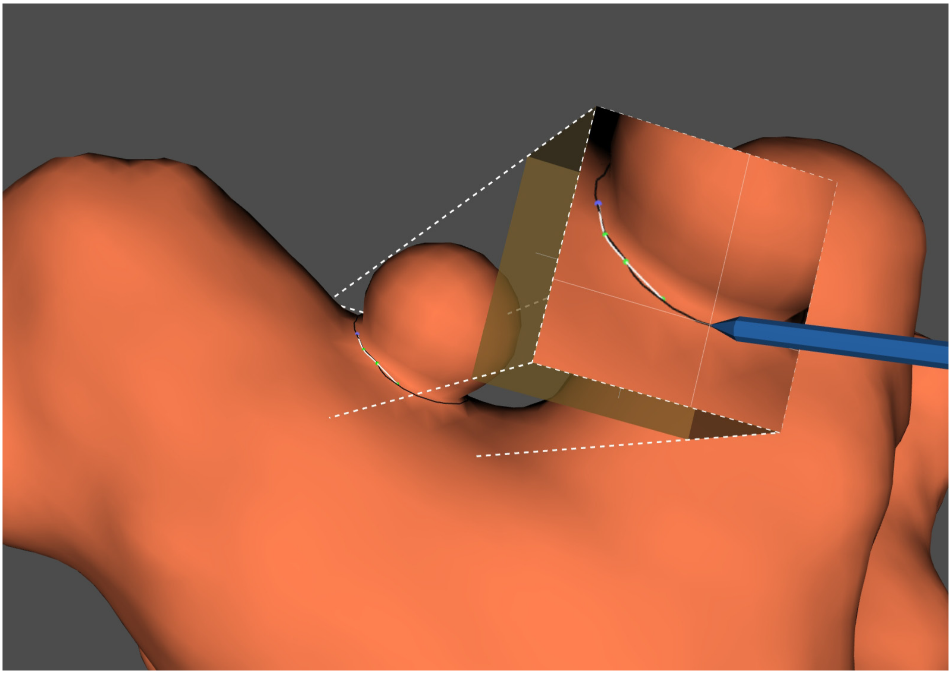


Figure 6.5: Screen capture of the tangible Magic Lens. The white dashed lines show which part of the scene the lens is currently showing.

The main purpose of the lens is not inspection but interaction. All 3D widgets are placed both in the main scene and in the Magic Lens view, and any manipulation of a widget in the Magic Lens projection is also instantly visible in the main view. Widgets are manipulated using the cube as a tangible controller; a second input device is used to select a location on this cube. This location on the actual device matches the corresponding location on the virtual representation of the cube. This location of the second input controller is then translated to a location on the texture used for the Magic Lens, and interaction takes place by emulating mouse input on the off-screen window. When the two devices touch a translucent white cross-hair cursor is shown in the lens, extending to the edges of the cube and centered on the input location, serving to facilitate selection of the desired location.

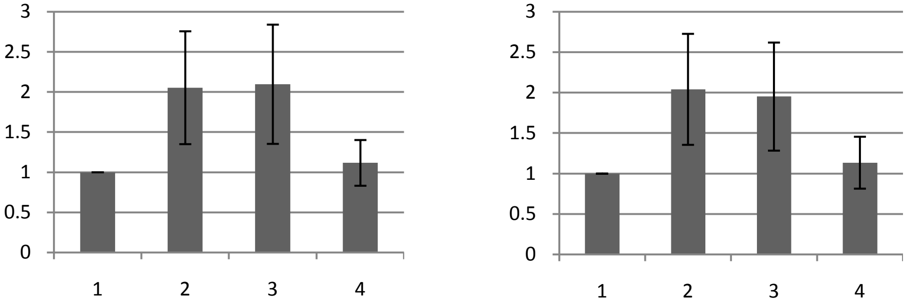
Since the interaction sends mouse commands to the off-screen window in principle any kind of interaction that is possible in an on-screen window with a mouse can also be performed in the same way on the Magic Lens projection using the tangible controller.

We have chosen to use a second 6-DOF pen device to select a location on the cube by using this pen as a 2-DOF device on the surface of the cube, but the same could be achieved using any (two-handed) 8-DOF controller, for example if a 6-DOF input sensor would be attached to a digital graphics tablet.

6.4.4 Results

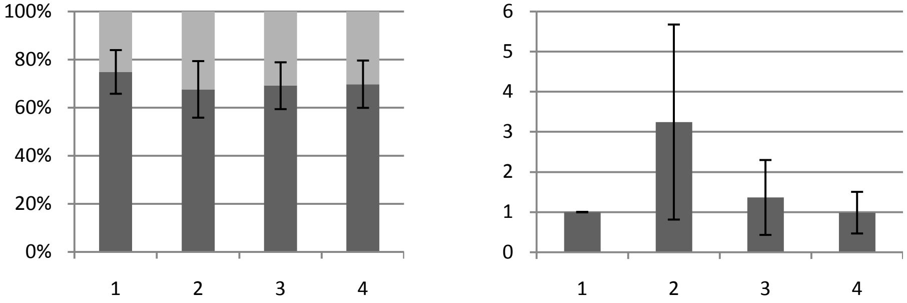
We conducted the experiment with 11 users with various levels of experience with 3D input devices. The users completed 7 trials for every method. The practice time before each method ranged from 1 to 10 minutes depending on the user and the particular method.

We assumed and also observed that the contours for the different spheres cannot be drawn in the same time and with the same accuracy even by the same users and when using the same method, and that some users are more meticulous than others. To compare all the different trials all results for a particular user and sphere are normalized with respect to the results when using method 1, which therefore always have a value of 1. Also an analysis of variance (ANOVA) was performed on the data.



(a) Relative contour drawing times with standard deviation for the four methods.

(b) Relative trial times with standard deviation for the four methods.



(c) The contour drawing time (bottom bar) as a percentage of the total trial time.

(d) Relative accuracy of each method with standard deviation.

Figure 6.6: Graphs of the measurement results. All four methods are shown relative to method 1, with standard deviation.

The average contour drawing time for each of the methods, relative to method 1, is shown in Figure 6.6a. The difference in time between method 2 and method 3 is not significant, while all other differences are significant ($p < 0.01$).

The average trial time for the four methods, relative to method 1, is shown in Figure 6.6b. The difference in time between method 2 and method 3 is not significant, while all other differences are significant ($p < 0.01$).

The contour time as a percentage of the total trial time is shown in Figure 6.6c. The difference in mean trial time percentage between method 1 and the other methods is significant ($p < 0.01$), while all differences between methods 2, 3 and 4 are not significant.

The accuracy of each method relative to method 1 is shown in Figure 6.6d. The difference between method 1 and method 4 is not significant, while all other differences between the methods are significant ($p < 0.01$).

All users preferred the methods using the mouse with and without the cube over the methods using only 3D devices. All but two users preferred using the mouse with the cube over using only the mouse, and all but one user preferred using the tangible Magic Lens cube over using the freehand technique.

6.4.5 Discussion

The accuracy and precision of the 3D tracking system is an important issue. Especially the tangible input controller used in method 3 relies on a correct calibration between the pen and the cube, as the on-screen representation of these devices needs to exactly match their relative positions and orientations; otherwise it is not possible to draw on the cube. The standard 2D mouse has a higher accuracy than the 3D input device. On the other hand 3D devices are more efficient for manipulating the world or 3D objects. Many novice users are also accustomed to using a mouse, while they find the 3D devices difficult to use.

The most preferred method was using the 2D mouse with the 3D world in hand, while using only the mouse was preferred over both 3D-only methods. Of these methods, the Magic Lens method was considered easier to use than the freehand drawing method.

The results show that the methods using the mouse are more accurate and certainly faster than the 3D input methods. Combining the mouse with 3D camera control is actually slightly slower than using the mouse alone, even though most users preferred this method. The users perceived the 6-DOF camera control of this method as an advantage over using only the mouse for this purpose, although the measurements do not support their view.

The tangible controller of method 3 is as fast as the freehand input of method 2, but it is significantly more accurate. This may be the result of a better sense of where on the surface of the object manipulation is taking place, or because the movement of the pen on the cube is initiated from the wrist as opposed to the whole arm with the unrestricted controller, which causes the movement to be more precise. In addition the pen rests on the cube, while method 3 requires constant muscle tension in the arm to keep the pen in a steady position.

The fact that in method 1 the fraction of the trial time spent during the drawing phase is higher, and thus that the initial viewpoint selection took less time, seems to imply that viewpoint selection using a 6-DOF device is somehow slower than viewpoint selection with the mouse. The probable cause for this difference is that in method 1 zooming the camera always occurs relative to a point that was previously selected with the ‘F’-key or relative to the center of the object if no point was selected, while for the other methods the cube always has to be actively moved into the desired center location for the zooming to have the desired effect, even if the desired location was the same one as during the previous zoom operation.

Using a tangible controller is much more accurate and easier than freehand drawing in empty 3D space. However, it requires a more accurate and much better calibrated 6-DOF input system than pure 3D freehand drawing.

6.5 Tangible Interaction Techniques

Interaction consists of using a tool on an object. With tangible props the prop is the object and a stylus is used as the tool, while the application determines how to respond to the interaction.

Physical interaction with a prop is performed when the tip of the stylus is touching the surface of the prop.

The shape of the prop P is described by its geometry, G_{prop}, which is simply the surface of P. The euclidean distance of a point x to this surface is given by $d(G_{prop}, x)$ and the actual surface consists of all points x where $d(G_{prop}, x) = 0$. The position and orientation of the prop P relative to any origin O are defined by the transformation function $T_{prop}(x) = (x')$. Thus given G_{prop} and T_{prop} we can determine where the surface is located relative to the origin O, or whether a point x relative to the origin O is located on the surface, which is the case if the distance is equal to zero, i.e., $d(T_{prop}(G_{prop}), x) = 0$.

The stylus also has a surface geometry, G_{stylus}, but the only essential part of it is the tip of the stylus S, which is a point x in space. The orientation and position of the stylus in space, given by the transformation T_{stylus}, simply provides a new value of x for S.

If the tip is touching the surface of the prop, then $d(G_{prop}, S) = 0$. If $d(G_{prop}, S) \neq 0$, then the tip is not touching the surface. The state of the interaction is thus only defined by the value of the function $d(G_{prop}, S)$.

Virtual interaction in a purely mathematical manner is performed with a point on a surface. The interaction takes place when the point is coincident with the surface.

The shape of the virtual prop P' is described by a geometry G'. This geometry is a mesh of vertices that define polygons, along with associated normal vectors. The position and orientation are also defined by a transformation function T'. The virtual stylus tip location S' is a point.

Again, the function $d(T'(G'), S')$ determines the state of the interaction, if $d(T'(G'), S') = 0$ then the virtual tip is touching the surface, and if $d(T'(G'), S') \neq 0$ then it is not touching the surface.

6.5.1 Virtual-Physical Interface

Difficulties arise when a physical and virtual interface are combined. The user interacts with a physical prop and stylus. She sees and feels these objects, she interacts by manipulating them and she receives visual and haptic feedback from them. Her notion of what she is doing is based on these physical objects, i.e., on T_{prop} and T_{stylus}. The computer tracks the positions and orientations of sensors or markers attached to these objects, and using this data it positions P' and S' accordingly. Thus the virtual system has its own notion of the interaction, using T'_{prop} and T'_{stylus}.

In theory, if the physical prop is a perfect match to the virtual model and if the tracking system delivers perfect accuracy, then the virtual interaction methods can be directly used to let the computer determine the state of the physical interaction, since $T_{prop} = T'_{prop}$ and $T_{stylus} = T'_{stylus}$. Therefore both user and computer will have the same notion of what interaction is taking place.

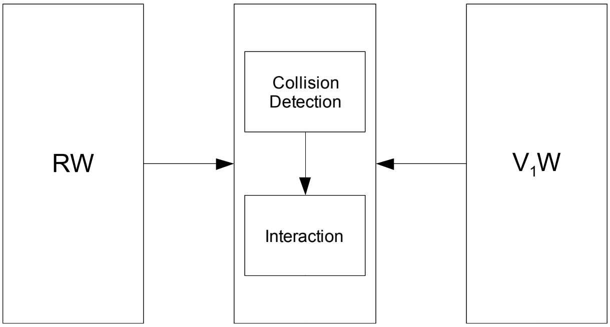


Figure 6.7: Interaction through collision detection. Real-world data from the tracker is combined with the calibrated world to perform correct collision detection.

Due to imperfections a number of issues arise, and the best possible result is that the computer will only have an approximation of the notion that the user has. The issues lead to changes in the transformation functions T', such that $T \neq T'$.

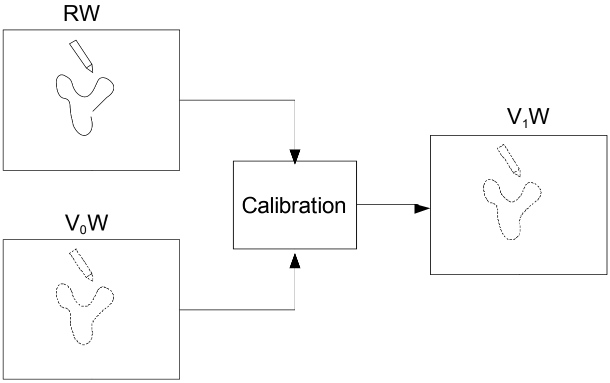


Figure 6.8: Obtaining the corrected model. The real-world data is combined with the original model to obtain the corrected model.

The new transformation T' is a concatenation of a number of transformations. In addition to the basic transformations of P' and S', corresponding to the real-world transformations T, there are a number of transformations that are necessary to bridge the differences between the real world and virtual world. Some of these transformations are static and some are dynamic, changing as the devices move. Most have to be found through calibration. Figure 6.8 shows how the real world RW and the original virtual world V_0W are combined and yield V_1W, the virtual world that matches RW. V_1W is then used to detect interaction, as shown in Figure 6.7.

While the added transformations reduce the differences between the physical and virtual world, there will always remain some variation. This variation can be quantified in the form of a number of tolerances ε_i, pertaining to both S and G. In effect the question is no longer whether S is coincident with G, but whether the distance between S and G is smaller than the tolerances, $d(S,G) < \sum\varepsilon_i$.

6.5.2 Interaction Techniques

When using a single prop the possible interaction will be limited to orientation and positioning of the prop and the corresponding visualizations. However, using a second device in conjunction with the prop enables the use of various interaction techniques, as locations on the prop can be used for interaction.

- **Selection**

A pointing device, such as a stylus, can be used to interact with points on the surface. The basic tasks that can then be performed are selection and dragging.

A single point on the prop can be selected, either directly by using the closest point on the surface or by shooting a ray from the pointing device and intersecting this with the surface. The exact selected point in the can then be used, for example for probing a value, or the point can be used to select larger parts of the data by using feature extraction and finding the feature corresponding to the selected point. Points can also be dragged, after which either the whole path or the start and end locations can be used, for example to indicate a direction.

- **Surface as proxy**

In addition to using the prop to interact directly with (a point on) the surface, it is possible to use the surface as a proxy for an alternative representation of the data. The operations are then carried out on the alternative representation instead of on the surface of the virtual prop.

If features are identified in the data that can be uniquely correlated with sections of or point on the surface, a selection operation anywhere on the surface will select the feature correlated with that part of the surface. For example selecting any point on the wheel of a car selects the whole wheel.

If a mapping can be created that surjectively maps the surface of the virtual prop to an alternative representation it is possible to perform operations on that alternative representation by interacting with the corresponding part of the surface. A simple example of this is a second iso-surface in the dataset created at a different isovalue and located below the surface used for the printed prop.

A point on a morphological skeleton, residing inside the prop, can in most cases be selected by mapping it to the closest points on the surface. Selecting a point anywhere on the surface then selects the corresponding point on that skeleton, and by dragging across the surface, the point is dragged to a different location on the skeleton, again corresponding to where the surface is touched by the stylus.

As an alternative, any point on the inside of the prop could be reached by first positioning a plane inside the prop and subsequently selecting a point on that plane by selecting a corresponding point on the surface of the prop.

- **Drawing**

Dragging can also be used for drawing and writing on objects. The dragged path can be recorded and shown as lines on the prop. In this manner it is possible to draw symbols or to write on the prop. If combined with handwriting recognition this is a powerful tool.

In the same manner a selection loop (lasso tool) can be created. For this the first and the last recorded points are connected to form a closed loop. A possible use is to cut away or to make translucent the selected part of the surface to show some underlying structure.

- **Buttonless interaction**

Given sufficient accuracy of the tracking system and the calibration of the props, it is possible to detect collisions between a prop and another tracked device, such as a stylus, or collisions between two props. The detection of the collision can then be used as an interaction trigger, thus enabling the start and end of interaction without requiring a separate button press. This is the most common way of using tools in the real world: the tool is used by holding it against an object. In addition this has the advantage that there is instant passive haptic feedback on the start and the current state of the interaction, as the user can feel when interaction is taking place.

The collision detection works by calculating the distance between the prop and the hotspot of the second device. The distance threshold for registering as a hit should be adaptive and must take into account the accuracy of the tracking system, especially the aforementioned orientation accuracy. If the accuracy of the tracking system is not constant, a map of the accuracy can be created and included in the collision detection calculation.

6.5.3 Sources of Errors

The implementation of usable tangible interaction requires high accuracy. Calibration is used to determine the differences between the virtual model and the prop, as well as differences resulting from limitations in various parts of the VR system. These issues are summarized in Table 6.1 and depicted in Figure 6.9.

The first issue is that any floating point computation has limited accuracy. This introduces ε_c as a suitable tolerance value that compensates for the accuracy of the computation. This value is usually chosen as a fraction of the size of the model or as a fraction of the size of the polygons in the model. Given the high accuracy of floating-point computation ε_c is typically a very small value, e.g., 10^{-5}.

Another issue is that the physical prop is not an exact reproduction of the virtual model. When creating a physical object from a digital model, there are always limitations with respect to resolution, manufacturing tolerances, artifacts, postprocessing, and other possible causes of discrepancies between the digital input and physical output. The result of these differences is that $G_{prop} \neq G'_{prop}$.

As an example, most casts shrink when the liquid cools and/or solidifies. This is compensated by making the casting molds slightly oversized according to the expected amount of shrinkage, but this will always result in a slight difference between the original model and the cast.

It is possible that the manufacturing inaccuracy is less than the resolution of the tracking system. However, in the general case there will be discrepancies that can be detected using the tracking system. These can be quantified through calibration. The result of the calibration is a transform T_m that translates between coordinates on the physical prop and the corresponding coordinates on the virtual model. There is a corresponding tolerance ε_m to compensate for inaccurate or absent calibration. The precision of 3D printing is typically at least 0.2mm; this also means that larger props will be relatively more accurate.

A number of issues stem from the fact that input devices do not have infinite accuracy. Mechanical and electronic properties limit the accuracy in various ways. Also, tracking systems report the position and orientation of the marker or sensor, but not where it is attached to the prop. The tracking space itself might not be linear but distorted. Finally it is possible that tracking information for the different devices is obtained on different moments in time.

- **Sensor location and orientation to model/prop origin**

The origin of a tracking sensor or marker will usually not coincide with the origin of the prop. This leads to a transform T_{sloc}, which translates between the sensor and object origin. Thus given the location and orientation of the sensor, the location and orientation of the prop is known.

The sensor can be attached to a prop at an ad-hoc location, or it can be affixed at a predetermined location, which can be either some convenient location on the data, or a special attachment point, which is not part of the data but is purpose-created on the virtual model and manufactured together with the prop. However, even in that last case there is some possible mechanical tolerance between the supposed and actual location of the sensor, ε_{sloc}.

If the attachment point is ad-hoc then T_{sloc} will always have to be determined through calibration. A simple procedure results in a large ε_{sloc}, while an elaborate procedure could render the tolerance insignificant. The same elaborate procedure can also be used to improve T_{sloc} to reduce ε_{sloc} when the sensor location is predetermined. The impact of this error will depend on the accuracy of the tracking system, and will generally be of a similar magnitude.

- **Limits of position and sensor accuracy**

The accuracy of the position and orientation as reported by tracking systems has specified limits. The positional accuracy can only be accounted for by introducing a corresponding tolerance ε_{track}, which can be taken from the published specifications of the tracking system or from actual measurements.

Orientation accuracy poses a special problem. The uncertainty about the orientation of a prop introduces uncertainty about the location of the surface that increases with the distance from the location of the sensor. This can be imagined as if a cone were used to cut out a portion of a sphere located at the sensor origin, with the tip of the cone also situated at the center of the sphere. Grossly exaggerated, if a tracking system would have an orientation accuracy of 90 degrees, the location of a point on the surface of the prop may actually be anywhere on the surface of a hemisphere centered at origin and of a radius equal to the distance between the origin and the point on the surface. Clearly no useful tracking system has such a lack of accuracy, but even in highly accurate systems this becomes an issue with average-sized props.

For interaction purposes this introduces a dynamic directional tolerance ε_{orient} determined by the location of the prop sensor and the location of the stylus tip S, directed perpendicular to the vector between the sensor and S.

- **Distortion of tracking space**

The tracked space is susceptible to distortion. This means that a linear change in the world position or orientation of a device may not always result in the same change in the position and orientation as reported by the tracking system. Since only a single point is tracked, and not every point of the surface of P, the distortion must be corrected to determine whether S_{tip} is located on the surface of P.

- **Tracking synchronization**

When using multiple devices it is possible that not all device positions and orientations are reported or sampled at the same time. This can be the case if one tracking system is used to track the prop while a different system is used to track the stylus, or if the tracking system does not sample devices simultaneously. When the location of P is known, S might already have moved, and vice versa.

When a new tracking report arrives for a device, prediction can be used to approximate the current position of the other device, which yields the transform T_{lat} for that device. Since prediction is always an approximation, a small additional tolerance ε_{lat} is introduced to account for the remaining error. This value could be extracted from the prediction calculations.

It is also possible to omit real prediction, and instead interpolate the current and previous locations of the newly reported device to match the approximate location at the time when the location of the ‘older’ device was reported. Since precise interaction is not likely to consist of fast movements this is an acceptable alternative.

Problem	**Transform**	**Tolerance**	**Significance**
Computational accuracy	-	ε_c	very small
Differences between model and prop	T_m	ε_m	small
Location of sensor/marker	T_{sloc}	ε_{sloc}	small
Tracking position accuracy	-	ε_{track}	small/medium
Tracking orientation accuracy	-	ε_{orient}	medium
Distortion of tracking space	T_d	ε_d	large
Latency	T_{lat}	ε_{lat}	medium/large

Table 6.1: Summary of error sources.

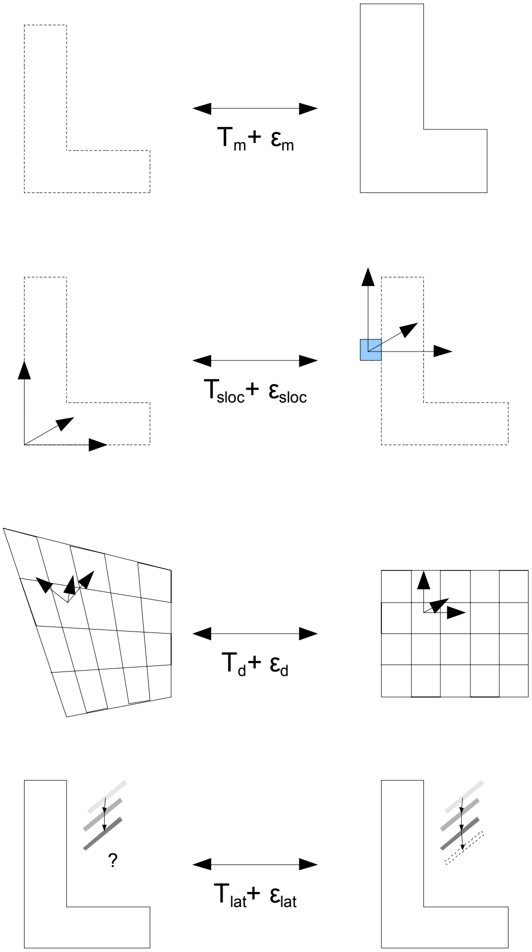

$T_m + \varepsilon_m$
$T_{sloc} + \varepsilon_{sloc}$
$T_d + \varepsilon_d$
?
$T_{lat} + \varepsilon_{lat}$

Figure 6.9: An overview of the transformations and tolerances.

6.5.4 Tangible System

Contact between the stylus and the prop is detected through intersection of the stylus tip with the surface of the prop. The intersection of this point with (one of) the polygons of the prop model, S' ∩ G', will yield a point if S' is located on G' (i.e., if the tip of the stylus is touching the surface). If the intersection yields an empty result then S' is not in contact with G'.

The uncertainty makes it impossible to use a single point as S. The location of S is not entirely certain, and neither is the location of the prop surface G. Instead of a single point, S is defined as an area around the tip of the stylus. This area is then intersected with G, and again a non-empty intersection, S ∩ G ≠ ∅, indicates that contact is made.

The size and shape of the area S is directly related to the sum of the tolerances pertaining both to S and to G. Since the interaction depends on the relative positions of S and G, it makes no difference mathematically whether the uncertainty about G is applied to G or to S.

There are interdependencies between some transformations and tolerances. A large tolerance can be used to compensate for an inaccurate transformation, such as a quick calibration resulting in approximate values. An elaborate time-consuming procedure could conversely be used to obtain a very precise transformation that is accurate enough to make the associated tolerance insignificant enough to be ignored altogether.

- **Stylus**

The transform T_{sloc}, translating between sensor and prop coordinates, can also be determined for the stylus. The origin of the stylus is normally the tip, or hotspot. The sensor is never located at this hotspot, thus the transform T_{sloc} must be determined to make meaningful use of the stylus. In some systems the stylus is factory-calibrated such that the tracking system already returns locations translated through the factory-determined T_{sloc}. However, the factory default might contain a slight error; therefore even these styluses should be calibrated.

The procedure for calibrating a stylus is straightforward (ref). The tip of the stylus is placed in a fixed position, for example a small dimple or hole drilled into a suitable fixed surface such as a table. The opposite end of the stylus is then moved around. Since the tip remains in a fixed location, the motion of the opposite end, and thus the sensor, will be constrained to the surface of a sphere. The recorded positions and orientations can then be used to find the location of the tip relative to the sensor, for example using least-squares fitting of potential T_{sloc} transforms to find the transform that creates the smallest spread when the sensor locations are transformed through it.

- **Prop**

The prop is calibrated in a number of steps. The procedure produces a combined transform comprising both T_{sloc} and T_m.

Initially, a number of calibration points are defined on the virtual prop. The corresponding points on the physical prop are marked. Then the stylus is used to indicate these points on the physical prop, and these locations are stored relative to the prop sensor. The transformation between the expected and actual locations of these points is then calculated. The virtual prop is then modified using this initial transform.

Next, a number of points on the surface of the prop are indicated. Together with the points from the previous step these are used to calculate an Iterative Closest Points transform between these points and the surface of the transformed virtual prop. The prop is again transformed using the result from the ICP calculation.

Finally, a number of points are again indicated on the surface of the prop, and this time the offset vectors between the measured locations and the surface of the virtual prop are used to warp the virtual model.

- **Tracking space distortion**

It is possible to map the distortion by systematically moving a sensor along a grid in the tracked space [90]. It is then possible to determine how much the reported locations differ from the grid locations.

The distortion can be static or dynamic. For example, in magnetic tracking systems metal objects cause distortion of the magnetic field and thus of the reported position and orientation of a device. This kind of static distortion can be mapped and compensated by inverting the discovered distortion transform, which yields the transform T_d. Small dynamic distortion may best be compensated by a tolerance ε_d.

- **Latency/sync**

Inevitably there will be a delay between the immediate haptic feedback of the user manipulating a prop and the visual response of the system. If the latency is high, this causes the user to slow down. In augmented reality systems in which the user has a direct view of the interaction area (e.g., not video-based) there is a visual latency between the movement of the physical props and the virtual additions. In video-based augmented reality systems on the other hand there is latency between the immediate haptic feedback from the prop and the corresponding visual feedback from the augmented video, but there is no latency between the real and augmented parts of the scene.

Various experiments showed that detection of the presence and the amount of latency does not follow Weber's law. This means that in a system with latency the smallest amount of additional latency detectable by a user is not a fixed fraction of the base latency, but rather a constant amount. This indicates that people adapt to and compensate for the latency in a system. This compensation does cause the user to perform actions more slowly in order to remain accurate. Research indicates that 16ms of visual latency can be noticed by users, and 50ms of latency in the visual system leads to deterioration of performance. Latency in haptic feedback affects user performance already at 25ms. However, for performing simple tasks for which only one type of feedback (visual or haptic) is sufficient, the brain is able to ignore the more delayed feedback and focus on the one which is faster.

Using tangible props for interaction reduces the effects of latency, since there is no haptic or visual delay from the 'representations' of the input devices when manipulating the props. The only latency is from the delay between an interaction event and the system response to the event. Keeping this in mind, user interaction should be designed in such a way that user dependency on computer feedback while performing a task is minimized.

- **Example accuracy of tracking system**

Even a perfectly matching printed prop will be useless if the tracking system lacks sufficient accuracy. The required accuracy of the tracking system depends on the intended use, but also on the size of the printed prop. For augmented reality systems the tracking should be at least accurate enough to register the prop with the augmentation within one pixel.

If the prop is used in combination with a second input device, the accuracy must be sufficient to determine which location on the prop is indicated with the second device. This in turn depends on the resolution of the original data, the size of the prop, and the size of the smallest relevant feature of interest. For example, in an educational molecular application it could be sufficient if the system can determine which atom or link between atoms is indicated on the prop. However, a precise measurement application might need an accuracy of one millimeter. To reliably detect collisions between props an accuracy of less than one millimeter is necessary.

An important issue is the accuracy with which the tracking system can determine the orientation of the prop, especially if the prop is large and the required precision is very high. Errors in the reported orientation lead to uncertainty about the exact location of the surface of the prop, and this uncertainty quickly becomes more significant than the positional uncertainty of the tracking system. For example, the Polhemus Fastrak magnetic tracking system is stated to have a maximum positional accuracy of 0.038mm and an orientation accuracy of 0.15 degrees. This means that for each centimeter away from the sensor the positional uncertainty increases by 0.026mm, thus already 1.5cm from the sensor there is more uncertainty about the position of the surface of the tracked prop stemming from an uncertain orientation than from an uncertain position. During typical use of the system the accuracy will be lower, thus the effect will be greater.

Tracking systems can suffer from distortion of the tracked volume. If this distortion is constant at certain locations, a distortion map can be created to compensate the error.

6.6 Integration with Augmented Reality

Augmented reality systems combine the real world with computer visualizations. Either existing objects are augmented with additional visualizations, such as annotations or measured data, or abstract physical objects are used as handles to manipulate virtual objects. The use of tangible props in such systems makes interaction with virtual objects as natural as interaction with real objects.

6.6.1 Types of Augmented Systems

The real and virtual scene can be combined in a number of different ways, each of which has specific issues.

- **Video**

Video based augmented reality systems use cameras to capture the real world and insert visualizations directly into the video stream. The user watches this video feed and sees physical objects along with virtual additions. A head-mounted display (HMD) or a handheld device can be used to achieve co-location and mobility. However, it is also possible to watch the scene on a separate screen if co-location and mobility are not needed. This is the simplest method of augmenting reality. There will inevitably be latency between a user action and the corresponding visual feedback due to video delay.

- **Mirror/transparent display**

Here a semi-transparent mirror is situated in between the user's head and his hands in the interaction area. A display, suspended somewhere above the mirror, is reflected in the mirror, on which the augmentation is rendered. The distance between the display and the mirror is chosen to match the distance between the mirror and the center of the interaction area behind it. This has the benefit that the focal distance of the user's hands or other items in the interaction area and the focal distance of the virtual scene are matched. The use of head-tracking is necessary for proper registration of the real and virtual scenes. Obviously, there is no delay from the real scene, but motion causes temporary misalignment between the real and virtual scenes.

Alternatively a projector can project the virtual scene directly onto the transparent mirror. Instead of a mirror it is also possible to use a transparent display. Recently demonstrated transparent OLED displays are very promising in this respect, as this significantly simplifies the setup. The disadvantage of such methods is that the focal distances of the virtual and real scenes are no longer matched.

There are also see-through HMDs that have the semi-transparent mirrors and displays integrated, with special optics changing the apparent focal distance of the virtual scene.

- **Projection**

One or more projectors are used to project texture augmentations onto physical objects, or even onto whole rooms filled with objects. The objects onto which the augmentation is projected can be simple geometric shapes, but also highly complex and detailed models. The objects can also be movable and tracked, in which case the projection is modified to follow the object as it is moved around. Multiple projectors can be used to make the augmentation visible from all sides. By using stereo projection and head tracking it is possible to add depth to the augmentation, for example making organs appear inside a dummy.

6.6.2 Handling Occlusion

An important issue in augmented reality systems is occlusion handling. Occlusion is the single most powerful depth cue and will override any other cue, such as stereo vision, focal depth or shadows. If the user's hands or other items are in front of a virtual object then obviously part of the object should not be visible. Similarly if a virtual object is in front of a real object, then part of that real object should not be seen by the user. If these rules are not observed proper depth perception is very difficult. Figure 6.10 shows an augmented reality video feed where occlusion is not properly handled.

In video-based systems occlusion can be performed using image segmentation of the video feed and knowledge of the shape and location of the props. Background subtraction can be used if the camera is stationary (and the background does not significantly change). It is however much more difficult to determine whether the user's hands are in front of or behind any part of the scene that only exists in the virtual world. Real objects can be occluded by virtual ones by simply rendering over the corresponding areas of the video feed.

In mirror or transparent display systems performing occlusion is much more difficult, as there is no direct control over or view of what the user sees. It is possible to use a number of cameras to determine the location of the user's hands and avoid rendering occluded parts of the scene. Occlusion of the user's hand in the mirror (or transparent display) can be reduced to a minor issue by reducing ambient lighting. Under such circumstances the already low contrast between the hands and the background will be reduced to almost zero when a bright overlay is visible in front of the hands. Another possibility is to use an LCD

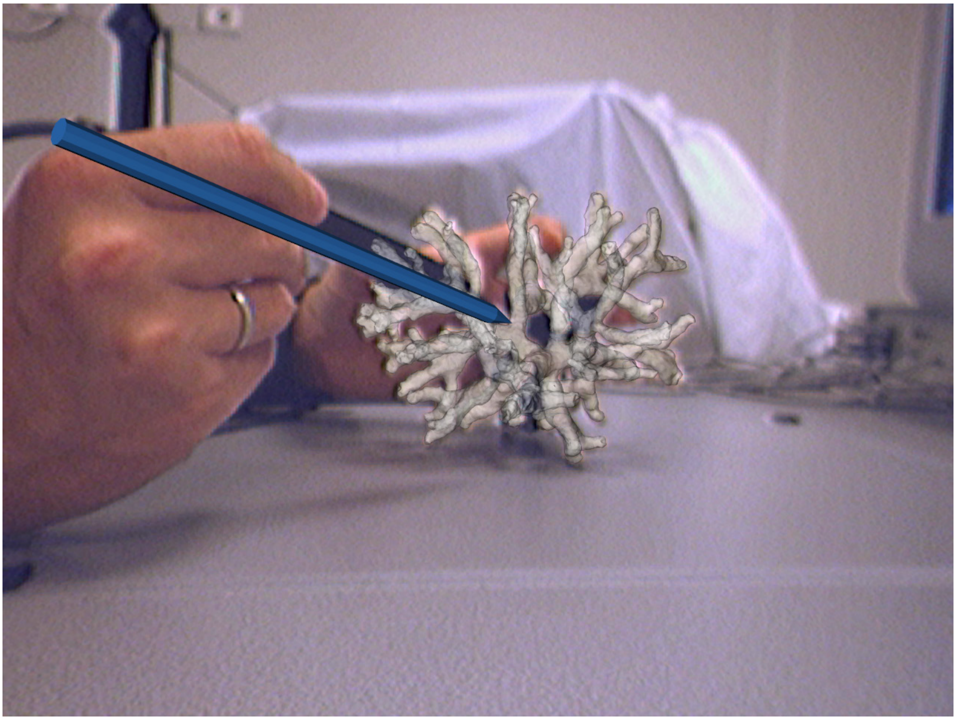


Figure 6.10: Improper occlusion effects in augmented reality using a video feed. The stylus is grasped in the hand, which occludes most of the stylus. However, the virtual representation of the stylus is not occluded at all and appears to be in front of the hand.

panel or other selective mirror technology to selectively make parts of the mirror or display completely opaque, thus occluding the real objects. More methods to handle occlusion can be found in [53].

In projection-based systems, occlusion of real objects by virtual objects that are not tracked is not feasible, since the occlusion would have to be projected directly onto the untracked object and this projection would need to be dependent on the viewpoint of the user. Also, many such systems are used to add only texture and color to physical three-dimensional models, in which case there simply are no occlusion issues that could arise.

Using printed tangible props reduces the effect of occlusion errors. Tangible props fully represent the extent of the virtual object. That way the hands are properly obscured by the prop itself. The occlusion of the augmentation of the prop by the user's hands can then be ignored, since a person handling and examining an (augmented) item will instinctively avoid blocking his view with his hands, just as he would when examining any ordinary object.

6.7 Tangible Corals

We have created a system for measuring various shape metrics of branching marine corals, which uses tangible input props for interaction in virtual and augmented reality. These props are three-dimensional prints of the surface of the coral, extracted from the segmented CT scans.

The system can be used both in an augmented reality mode and in a virtual reality mode. The augmented mode can be used to interact and to visualize results in the context of the original object. The virtual reality mode is useful for examining magnified visualizations of these complex structures.

The augmented reality mode was implemented using a video feed from a webcam. The coral and associated data were rendered on top of the video images, and this combined scene was shown to the user on the screen.

There are many reasons to prefer printed replicas over the original coral specimens. First of all, coral is very fragile and must be handled very delicately, and could easily break already during an attempt to attach a tracker. In addition many specimens are rather small, thus a larger than life size print would make interaction easier. Also, having several copies enables remote collaboration with tangible props.

6.7.1 Printed Coral

To experiment with tangible data props we have adapted our existing interactive system for measuring CT scans of marine coral and visualizing the results. In addition to CT data the system can measure data resulting from numeric simulation of coral growth. We have printed three specimens of marine corals scaled at 100% and 150% of the original size; enlarged prints make it easier to reach the inner regions of these complex objects, while at 150% enlargement it would still be possible to print all three props in a single batch, reducing the printing time and cost. The diameter of the original specimens ranged from 10cm to 15cm, the resulting props thus had diameters of 10cm to 22cm. A Polhemus Fastrak tracker sensor was attached as close as possible to the center of these props to minimize the effect of orientation uncertainty. Most of the props are shown in Figure 6.11, and another can be seen in Figure 6.12 with a sensor attached and used in conjunction with a tracked stylus.

The props have been printed using a Z Corporation Spectrum Z510 3D printer, which has a resolution of 600dpi in the X and Y directions and approximately 300dpi in the Z direction. The printer employs inkjet technology to inject a bonding agent into successive layers of fine powder, bonding the powder into a solid object. The powder particles are typically 0.05mm to 0.1mm in diameter. After the bonding agent is completely cured the printed object is removed from the powder and impregnated with epoxy to increase structural strength. This results in an object with a fine grainy surface texture, which can be sanded or polished to create a smooth surface. The possibility exists that the object is slightly deformed during curing, and sanding will certainly reduce the thickness.

The tracking system is a Polhemus Fastrak with a stylus and a standard sensor, which is attached to the prop. The accuracy of this system is claimed to be 0.038mm for position and 0.15 degrees for orientation. The sensors are sampled at a frequency of 60Hz, but are not being sampled simultaneously. The latency between motion of a sensor and display update has been determined to be approximately 60ms.

The coral props were calibrated using a two-step procedure. In the first step, an initial orientation and position of the prop is found. In the second step a more exact calibration is found using an iterative closest points approach.

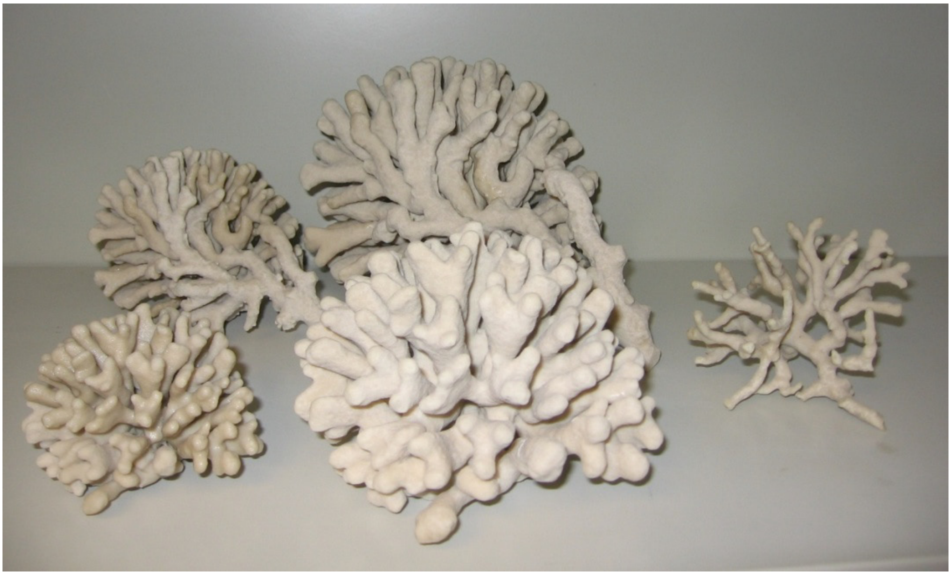


Figure 6.11: Tangible props printed from coral data. The front left prop has a diameter of 10cm.

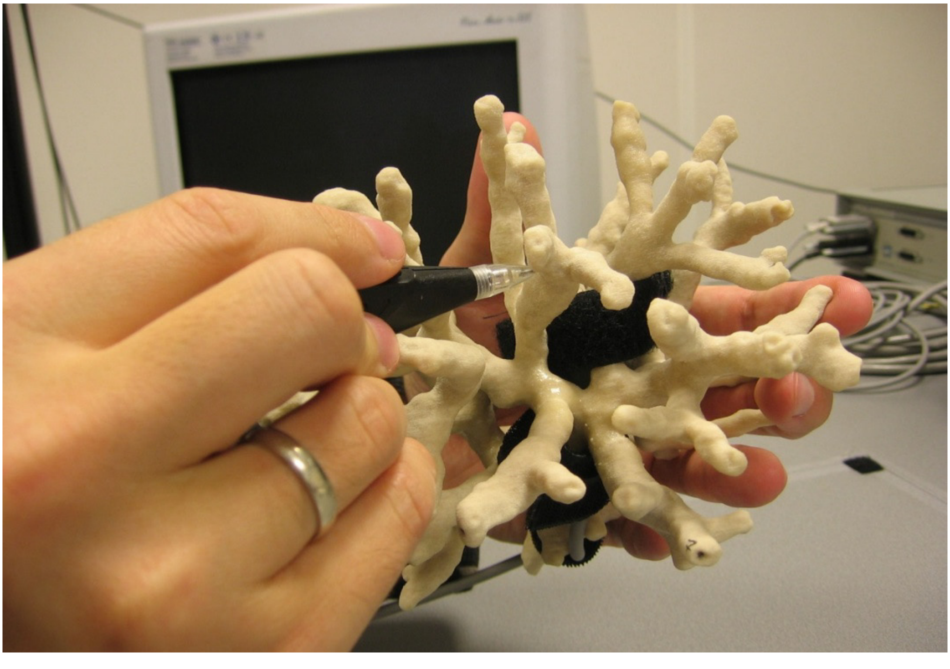


Figure 6.12: Tangible coral prop with attached sensor and stylus. The prop is held in the left hand, while the right hand operates the stylus to indicate locations on the surface.

A number of points (12) were defined on the surface of the model. For each point on the model the corresponding point on the printed prop was then indicated with the tip of the stylus and the button was pressed. The position of the stylus tip relative to the prop sensor was then recorded, using the current position of the stylus and the orientation and position of the prop sensor. After all the points were recorded the least-squares optimal transformation was computed that would transform the pre-defined points to the recorded points by using only rotation and translation.

This initial transform was then refined using an iterative closest points (ICP) method [68]. Random points on the surface of the prop were indicated with the stylus. These points were thus known to lie on the surface of the prop. These points were then used as source points in an iterative closest points search for an optimal transform from these points to the surface of the model (to which the initial transform was already applied). The model was used as the target data set for the ICP method.

Using this method we were able to calibrate one of the 150% scaled props to an average accuracy of approximately 0.7mm, which is approximately one third of the maximum attainable accuracy of the tracking system given the size of the prop, and about 1/13th of the diameter of the branches of the prop.

6.7.2 Measuring and Exploring

The application is used to measure various shape parameters of the coral specimens. The props are used on one hand to indicate locations of interest, and on the other hand to render the measurements in the context of the original object.

The original application was used to measure the whole coral specimen, and the tangible prop can be used in the augmented reality setup to explore these results. The tangible prop is also used to perform certain measures in indicated locations.

- **Local branch thickness**

The stylus is used to select a point on the surface of the coral. The thickness of the branch corresponding to the surface location is then measured at the indicated point and visualized both as an inscribed sphere of corresponding thickness and as a numerical value.

When the button is used to change interaction state, the measured location changes with the movement of the stylus while the button is pressed, and after it is released the last location remains visible.

In buttonless mode, the location changes as long as the stylus touches the surface. The last measured location is shown after there is no more contact with the surface and while no new location is touched.

- **Branch distance measurement**

The distance between specific locations on the branches can be measured interactively. The stylus is used to indicate two points on the surface between which the distance is to be measured. This is visualized as a line annotated with the corresponding value as a number.

With the button, the first location is selected on the surface when the button is pressed. A line is then drawn from the selected location to the tip of the stylus. When the button is released, the second location is selected. The measured distance is shown until a new starting point is selected.

In buttonless mode, the first time the surface is touched a point is placed at that location. As long as the surface is touched, this point moves along with the stylus. When the surface is no longer touched, the last contact location is used as the starting point, and a line is drawn between this point and the stylus tip. The second point is selected in the same way, and the final location of the second point is again the last location where the stylus touched the surface.

The branch measuring tool can be seen in Figure 6.13.

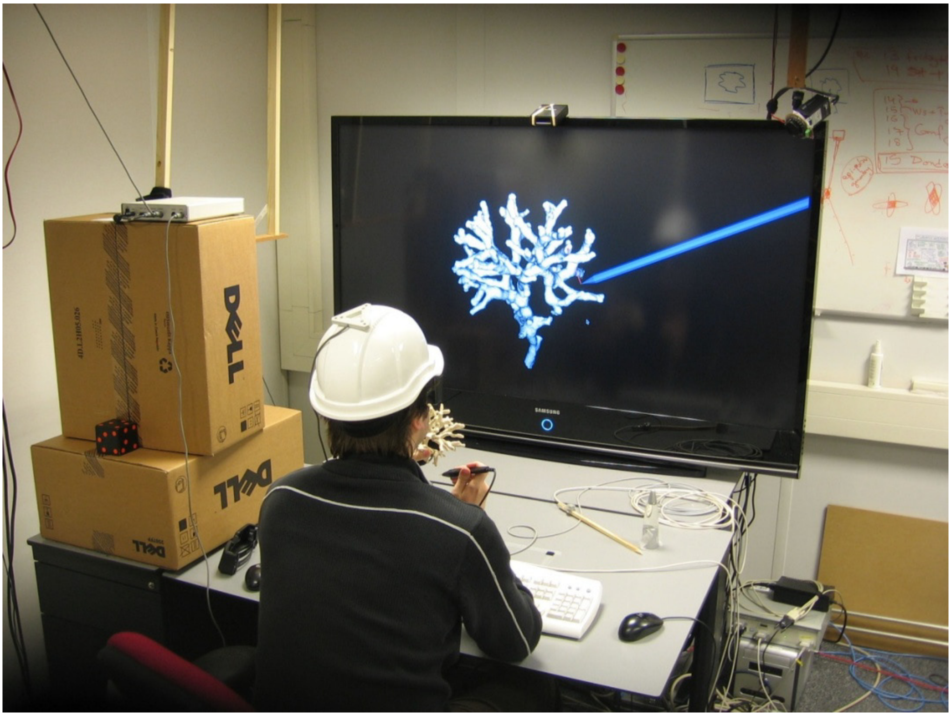


Figure 6.13: Interactive coral measurement application. The user interacts with the small prop in his hand, while a greatly enlarged virtual representation is seen on the large 67 inch stereoscopic display.

6.7.3 Usability

We have subjectively compared the usability of different types of input for our measuring system through a theoretical analysis. As a base for comparison we use an interface with a tracked stylus for pointing and a second sensor for positioning and orienting the data. This is compared to two interfaces using a tangible printed prop. The first uses the button on the stylus for signaling interaction, the second uses collisions between the stylus and the tangible prop as the trigger. We will refer to these methods as S, T and TB respectively.

Method S only provides visual feedback, which necessitates a tradeoff between speed and accuracy when using the method. This combined with the inherent latency makes it more difficult to use. However, the lack of dependency on any specific coral makes it very flexible. In addition it is trivial to implement, and it requires no calibration.

Methods T and TB provide latency-free passive haptic and visual feedback through the prop, which makes it easy to quickly interact with the desired location. However, for each new coral a new prop must be created, and switching data and props requires the calibration procedure to be repeated. Method TB has a small advantage over method T because it requires fewer actions to complete a task: both methods require the same motions to be performed, but in method TB these are not followed by a button press. The advantage of method T is that less accurate calibration is sufficient. These findings are shown in Table 6.2.

Aspect	**Standard (S)**	**Tangible (T)**	**Tangible Buttonless (TB)**
Speed	-	+	+
Accuracy	o	+	+
Ease of use	-	+	+
Feedback	-	o	+
Latency	-	+	+
Flexibility	+	o	-
Easy to implement	+	o	-
Calibration effort	+	-	-

Table 6.2: Findings of the subjective relative usability of three input methods: standard two-handed non-tangible input (S), input with a tangible prop using a button as an interaction trigger (T), and input with a tangible prop using contact between the stylus and the surface as the trigger (TB). From top to bottom are different aspects of the methods. A ‘+’ indicates the method is relatively good in that aspect, an ‘o’ indicates neither good nor bad, and a ‘-’ indicates the method is relatively bad in that aspect.

6.8 Conclusion

In this chapter, we have explored the use of tangible props as input devices to provide a graspable physical representation of data. This allows the use of the sense of touch when interacting with scientific visualizations in virtual and augmented reality systems. We have explained the concept and applicability of tangible props, and motivated the advantage of using tangible input with a user study. We considered the technical requirements of systems that use such props, and presented an application using tangible printed corals.

The interface that is thus created is much more natural to use than the ubiquitous standard pointing devices used in most systems. Instead of using input devices to merely remotely interact with virtual objects, with all the inherent latency issues, the input devices become the actual objects with which the user interacts, and which she can manipulate in the same natural manner as she would any other physical object. Thus in addition to performance benefits, the application of tangible props increases the sense of immersion in virtual and augmented reality environments.

Chapter 7 Conclusion and Future Work

Marine branching corals are a key part of marine ecosystems. The shapes of these complex objects are in part influenced by environmental factors. Quantitative analysis of their morphology provides insight into the mechanisms that shape the colonies, and gives an understanding of how their environment changed over time. This thesis has explored the techniques that are necessary to reliably perform shape analysis on CT scans of branching corals.

An interactive system for the quantitative analysis of the shape of branching coral morphology has been implemented. From CT scans a morphological skeleton is extracted, which is used to perform measurements on the coral; this process is both interactive and automated. The results can then be explored in a highly interactive environment. Augmented reality is used to provide a natural user interface, for which replicas of the coral are printed and used as tangible input props.

The research questions in Section 1.2.1 were formulated as:

- What is the effect of the sensitivity of skeleton extraction algorithms in relation to robust and accurate measurements of branching objects?

We extract morphological skeletons from the CT scans to quantify the morphology of the corals. As these skeletons are used to perform measurements, inevitably a difference in the skeleton causes a difference in the values resulting from the measurement. Noise and other artifacts are present in the data, and the sensitivity of the algorithm causes a different location or a different value to be measured. Another issue is that even given the same input data, different skeletonization algorithms produce different skeletons. For these reasons we have compared different algorithms by analyzing the resulting measurements.

- What is the role of 3D interaction and automation in the measurement process?

Interaction must be used when editing the data to repair anomalies. For automatic steps parameter settings are interactively assessed, and the results need to be verified. Steps that cannot be automated require interaction to be performed manually.

Interactive data exploration is a very useful tool for analyzing the results of the measurements, particularly the use of linked interactive 2D and 3D visualizations.

- What is the role of augmented reality and tangible interfaces for interactive shape analysis of branching objects?

The three-dimensional nature and branching complexity of stony corals can easily be disorienting. This is mitigated when using an easy and more natural interface for interaction with these objects. A tangible interface enables the use of the sense of touch for interaction. Augmented reality further serves to bring the actual interaction into the physical world.

7.1 Future work

There are several directions in which the described work can be continued, and where improvements can be made.

- **Skeletons**

Only a few skeletonization algorithms were compared in this work. Numerous very different algorithms exist, each with their own peculiarities that make them more or less suitable for a particular application. New algorithms also appear on a regular basis. A more comprehensive method of quantitative analysis of the performance of skeletonziation algorithms is needed. Instead of using application specific metrics, such a method would measure quantifiable characteristics of skeletons. Then, in order to select an algorithm for a particular application, each characteristic can be assigned a specific importance, depending on how that characteristic affects the application.

- **Graph Extraction**

The process of extracting the graph from 3D coral images can be further automated. In particular, an automatic method for disconnecting cycles in the graph would be an interesting challenge. This would likely involve locating the corallites and tracing them back to their origin, thus determining how exactly the coral grew into its final shape. When this information is determined, it is also likely that the root can be determined automatically. It can also conceivably result in a coral-specific skeletonization algorithm.

- **Uncertainty Visualization**

Only a single artistic rendering style was used for uncertainty visualization. The merits of different techniques should be investigated. Additionally, the usefulness of uncertainty visualization using a combination of more than one non-photorealistic technique with photorealistic rendering, and a combination of only non-photorealistic techniques could be examined. The combination of a technical drawing style with the sketchy style seems like a logical step, since a technical drawing style conveys exactness.

A better, hierarchical method for visualizing uncertainty in large line data using the hybrid rendering approach also warrants further research. In a close-up view the current method functions well, but in a more global view it might be more useful to visualize the average uncertainty of larger parts of the data.

A limiting factor in the use of rendering styles for visualizing any variable is the limited number of discernible levels. The use of a number of different styles may increase the number of different levels that can be discerned, but determining which style corresponds to which range of levels may be much more difficult than when using only two different styles.

- **Tangible Interaction**

The precise calibration of printed props is a time-consuming process. By using additional optical tracking this could likely be largely automated. Alternatively, manufacturing sensor mounts as part of the prop would greatly simplify the procedure.

An extension of the tangible printed interface would involve the use of modular props. With complex props consisting of a single piece some of the inner parts are hard to reach. This becomes much easier if parts of the prop can be removed. This could involve some means of keeping track of the current configuration of the prop, or using separate tracking for each constituent part.

Appendix A 3D input in VTK

To experiment with 3D input devices, we have added support for such devices to VTK [45,71]. This appendix describes the implementation of this VTK extension.

A.1 Input in VTK

Input in VTK is inherently mouse and keyboard oriented. The mouse may be used to modify the viewpoint, and 3D widgets are used to interact with data. However, these 3D widgets subscribe to 2D keyboard and mouse events, and react depending on the coordinates where the events occurred. There is no central location where 2D input is translated to 3D equivalents before being passed on to the 3D widgets or the camera.

The main event loop in VTK waits for events from the particular windowing system (X-Windows, Microsoft Windows, MacOS), and then dispatches these to interested objects, such as the window interaction class (vtkRenderWindowInteractor) and the widgets.

Widgets in VTK are separated into a representation part and an event processing part. The event handling part first receives specific input events from the mouse and keyboard (the left button has been pressed, the mouse has been moved, etc.) and translates these into widget events (initiate selection, motion performed, etc.) thus making the rest of the widget independent of any particular binding of buttons and/or keys to actions. The event handling part then in turn handles these widget events. Conceptually, initially the representation is first asked whether the specific screen location corresponds to a part of the representation on which actions can be performed, and if so, the interaction state of the representation is set to correspond to the specific action (for example translation). Subsequent (mouse) motion is passed to the representation, which then decides how to react to the motion, based on the interaction state. The event handling part can also change other widget-specific features of the representation in response to input events, for example constraining the motion of a handle to a particular axis when the shift key is held down during interaction. The event handling part has no need to know what the widget looks like on the screen, or even whether the widget is 3D or 2D. Many widgets have several representations to choose from.

The representation part handles the visualization of the widget. It also must tell the event handling part whether interaction is possible at a specific location (hotspot). It has a state that determines what happens when it is informed of cursor motion. If it is in a state in which interaction actually occurs, the representation part updates its own geometry according to how the mouse was moved, and how the specific representation is designed to respond to motion, including constraints on the possible motion. It is possible to create 2D widget representations that are placed on the 2D overlay layer on top of the 3D scene, in addition to regular 3D widgets that are placed in the virtual scene.

The widgets can be created in a somewhat hierarchical manner. A complex widget can consist of a number of other simpler widgets, with perhaps some additional geometry visually connecting the constituent widgets. Many widgets thus consist of one or more simple spherical handle widgets, which are attached to some additional geometry such as lines or planes. These handles can be dragged to different positions, and the representation ensures that the remaining geometry of the widget is updated accordingly. Often the additional geometry itself can also be manipulated, usually moving the widget as a whole or rotating it. Such a hierarchy does not consist of full, completely independent widgets. Rather, the parent widget takes over some of the event processing of the constituent widgets.

The event handling part of the widget can dispatch VTK events that are either specific to that widget, such as ‘last point for curve has been placed’, or events that are common to all widgets, such as StartInteractionEvent, InteractionEvent and EndInteractionEvent, which indicate respectively the start of an interaction sequence (e.g., a handle has been selected with the mouse), the change of the state of the widget (e.g., when the location of the handle has changed), and the end of an interaction sequence (e.g., the handle has been released).

A program that uses VTK widgets creates an instance of a widget and a suitable representation for the widget. It should then only respond to the high-level VTK widget events, and not to any mouse or keyboard events.

A.2 VRPN

VRPN, the Virtual Reality Peripheral Network [79], is a client-server framework for connecting to various input devices. The client is the application that wants to receive input events. It connects to a server, which polls the physical device. The client and server can run on separate computers, on the same computer, or even within the same process.

From the point of view of the application, there are only a few classes of devices. These include Button and Tracker devices, which provide access to buttons and 6-DOF devices respectively. There are also provisions for other types of input, as well as force-feedback, but these are outside the scope of this appendix.

The client only needs to know the classes of the devices it wishes to receive input from, as well as their names (which include the network address) and the number of buttons or sensors on the device. For 6-DOF devices the client also should know whether the device generates position and orientation, velocity or acceleration data, since most devices provide only one of these metrics. The specifics of dealing with any particular device, such as a Spaceball or a Polhemus, are handled by the server and are entirely hidden from the client application.

A.3 Our Approach

The first goal in our approach was to avoid any changes to the VTK source code. Instead, the whole 3D input mechanism has been created as a pure add-on extension. Also, existing VTK programs should be able to use 3D input without any modifications, or at least those programs that use only VTK widgets for interaction and do not respond to keyboard and mouse events directly.

The second goal was that it should be possible to explicitly use 3D input in all of the programming languages for which VTK wrappers are generated, most importantly Python. It should thus be possible to have full access to the 6-DOF tracking data, just like the wrapped languages have direct access to VTK mouse and keyboard events. This is useful to implement special functionality in the wrapped languages.

A.3.1 Input

The main VTK event loop is overridden to also check for VRPN events. These events are propagated as user-defined VTK events, and are ignored by VTK objects that have no knowledge of the 3D extension.

A.3.2 Extension Mechanisms

VTK contains a set of mechanisms that make it very easy to change the behavior of applications that use VTK without modifying the application itself.

The first mechanism is the use of the factory method design pattern. All objects in VTK are instantiated by using a static method that returns an instance of the requested object. This method may instead also return any subclass of the requested object, with a different behavior than the original class, and it also enables the ‘instantiation’ of abstract classes, with the factory deciding which particular subclass is appropriate; this feature is the preferred method to instantiate rendering-related classes in VTK, where instances of the abstract superclasses are requested, but VTK instantiates different subclasses that support the rendering API and windowing system that is currently available and/or configured. These overrides may be registered and changed at runtime.

The second mechanism is the use of a search path for loading additional VTK libraries. A search path can be specified using an environment variable. This path is then searched for dynamically linked libraries that support a specific VTK API, and all matching libraries are then loaded and initialized. In combination with the first mechanism this feature can be used to change the behavior of existing programs without any modification to the original code or binary, by supplying an automatically loading library that replaces the classes used by the program with classes that behave differently.

These two mechanisms can be used by our extension to add 3D input support to existing programs. When the library is auto-loaded, overrides are registered for the interactor and for the supported 3D widgets. When the extension is explicitly used, either directly in C++ code linked with the library or when used in a wrapped language, the override is not registered by default. However, in both cases the override mechanism is used to instantiate a specific tracker library interface object from an abstract base class, although in most cases VRPN is the only available option.

A.3.3 Tracker Communications

All communication with the tracking library is handled in a library-specific subclass of the vtkTracker class. All necessary external interface methods for configuration and communication with VTK are already defined in vtkTracker, thus the subclasses only need to implement library-specific parts. The rest of the system should not normally need to be aware of which tracker library interface is actually used.

The mapping of tracker library devices and sensor numbers to device numbers of the 3D extension is also implemented in this class. Configuration occurs by mapping VTK 3D device numbers (any integer >= 0) to (string, number)-pairs. The programmer must know how the tracker library interface interprets this data; the VRPN interface uses the string as the device name/address, and the number as the VRPN sensor number on that device.

An identical method is used to map button devices: a VTK 3D button number is mapped to a string and number, which for VRPN correspond to the device name/address and button number.

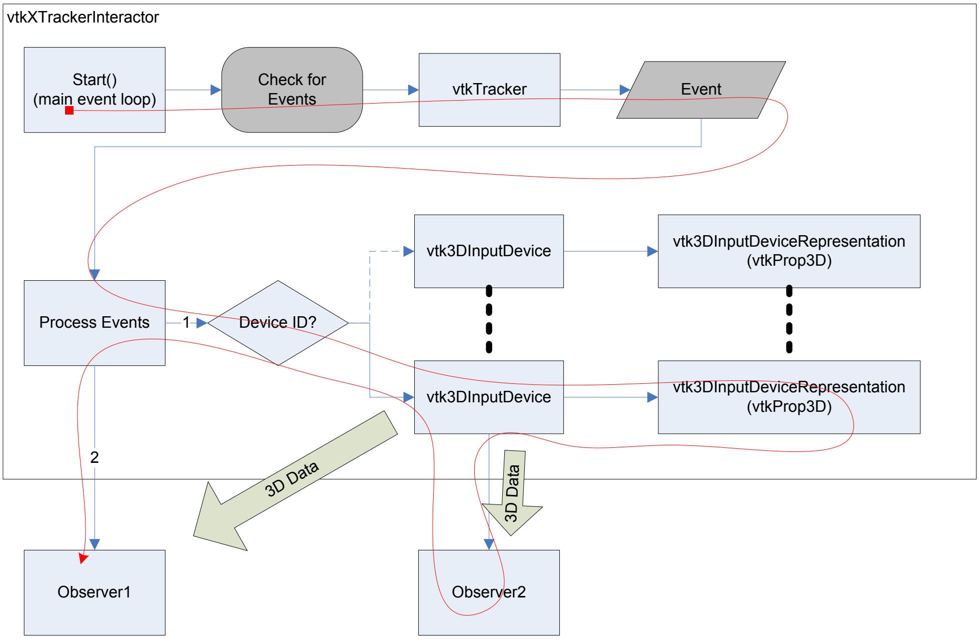


Figure A.1: The processing of events. The red line is the flow of control. The observers could be 3D aware widgets, or any user-supplied observer. Note that both observers get the actual 3D data (position and orientation) from the vtk3DInputDevice object.

A.3.4 Tracker Interactor

The 3D tracker interactor class vtkXTrackerInteractor holds and provides access to the information and other objects used by the 3D extension. It contains the vtkTracker object and a vtk3DInputDevice object for each configured input device. The interactor is a subclass of vtkXRenderWindowInteractor, the VTK interactor class specific for the X-Windows environment. In VTK the superclass vtkRenderWindowInteractor is the central class for processing user input, and if the 3D input were to be built into the core VTK, it would make more sense to put most of this functionality in the superclass instead, but this would then impact the core VTK API.

The vtkXTrackerInteractor class provides a new main event loop, which not only checks for X-Windows input events, but also instructs vtkTracker to check for tracker events. It then passes tracker events to the appropriate vtk3DInputDevice object for further processing, and finally signals a 3D motion event. Button events are handled similarly, except they do not pass through an additional vtk3DInputDevice object.

A.3.5 Event Propagation

The tracker interface object checks for tracker events. It processes them into a form suitable for further use, a 4x4 matrix for tracking events, a button ID for button events. Events indicating new tracker data are passed to the interactor, which in turn propagates them to the appropriate vtk3DInputDevice object. This object applies a number of transforms to the matrix, for example to reposition the device representations in the reference frame of the camera, then it sends an event indicating new data is available. After this, the interactor also sends an event indicating new data. An event observer has the choice to subscribe to the motion events either from the interactor or the appropriate vtk3DInputDevice object; the difference is that the interactor will send events for any device motion, while the vtk3DInputDevice object sends events only for the device to which it corresponds. The process is depicted in Figure A.1.

The button events are sent only by the interactor, as they do not require additional processing. This does however preclude the possibility to subscribe to events for only one specific button.

A.3.6 vtk3DInputDevice

The vtk3DInputDevice class processes the 4x4 matrix supplied by the vtkTracker class. The most important function is the translation from tracker (world) coordinates to a coordinate system relative to the camera position. This causes the device representations to always appear in the same position relative to the camera, regardless of where and how the camera is placed in the scene. This is used to achieve co-location between the physical devices and their on-screen representation, even when the camera is repositioned in the scene. This behavior can be disabled on a per-object (thus per-device) basis, but even when it is enabled it is possible to access the current device coordinates in tracker space.

The reasoning is that the tracker coordinate system is fixed relative to the display device space, while the latter is not fixed with respect to the virtual world space, as the camera can be positioned anywhere in the virtual world.

This behavior should be turned off when a head-mounted display is used. In this case the display is no longer fixed relative to the tracker coordinates, while it makes sense that the virtual world coordinates remain fixed relative to the tracker coordinates.

A.3.7 Device Representations

The on-screen representations of the input devices can be any subclass of vtkProp3D. This means that any regular VTK object that can be shown in the 3D scene can be used as a representation, for example a vtkActor or even a vtkVolume.

However, there is a special subclass, vtk3DInputDeviceRepresentation, which provides additional built-in functionality to change the representation analogous to how mouse cursor shapes are changed in VTK. Thus, when using these specialized representations, 3D-aware widgets can, for example, change the visible representation to indicate that the device is within range of a widget hotspot.

The 3D interactor automatically adds the representations to the renderer that is currently set as the active renderer for 3D interaction, and removes them when the active renderer is changed.

A.4 3D-aware VTK Widgets

Due to the requirement of avoiding any modifications of the VTK source code, for each 3D widget present in VTK a 3D-aware extension subclass must be created separately, rather than modifying the base widget class. Such modified widgets then subscribe to both 2D and 3D events from the interactor. 2D events are passed on to the original VTK widget code, while 3D events are handled separately.

As mentioned before, VTK widgets are split into two semi-independent parts. The communication between these parts is designed around 2D mouse event positions. However, this 2D coordinate communication means that, in order to implement 3D input support for a widget, both the event handling part and the representation part need to be modified in order to pass 3D positions to the representation part. This is achieved by creating a 3D-aware subclass of each part. Recall that the event handling part receives the mouse events from the interactor and translates these into the appropriate actions to be performed on the widget; the representation part is responsible for converting the 2D positions to 3D positions when and if necessary.

The 'communication protocol' between the event handling part and the representation consists of only a few methods, which are invoked on the representation:

- ComputeInteractionState(X, Y) tells the representation to compute an interaction state based on the given coordinates. This state is returned but usually also stored in the representation.
- GetInteractionState() and SetInteractionState() are used to query and change the current state.
- StartWidgetInteraction(X, Y) is typically called when a button is pressed.
- WidgetInteraction(X, Y) is typically called when the mouse is moved after interaction was initiated.
- EndWidgetInteraction(X, Y) is typically called when the button is released.

Note that it depends on the widget what the value of the state actually means, when it is set and by whom, and how it is used by the widget representation(s). This can actually vary wildly between widgets.

The modifications to the representation part in essence consist of duplicating the 2D-specific functionality of the representation, but with the use of 3D coordinates. Thus, where the 2D representation would calculate whether the mouse is within range of some part of the geometry, the 3D version needs to consider the distance between the 3D position of one of the 3D input devices and the geometry. Where the 2D version calculates how to move the 3D geometry when the mouse is moved to a new position, the 3D version can use the old and new 3D positions of the device.

In order to retain the original 2D functionality we have implemented this not by creating separate 3D methods to be called by the event handling part, but rather by overriding the original methods and keeping track of 1) whether the event causing the current call is triggered by a 3D device and 2) whether the interaction sequence was initiated by a 3D device (typically when a button associated with 3D interaction is pressed). This is accomplished by two corresponding Boolean flags. The first is set by the event handling part, the second is set by the representation itself when StartWidgetInteraction() is called and the first flag is set. For WidgetInteraction() and EndWidgetInteraction() 2D events are ignored when interaction is 3D and similarly the 3D events are ignored when interaction is 2D. If a 2D event is received in 2D interaction mode, the original method is called. In pseudocode, the representation's modified methods thus look like this:

```
if ( event_is_3D ) :
      if ( interaction_is_3D ) :
            do_3D_processing()
      else :
            do_nothing()
else : # NOT event_is_3D
      if ( NOT interaction_is_3D ) :
            original_2D_code()
      else :
            do_nothing()
```

The methods ComputeInteractionState() and StartWidgetInteraction() only use the first flag and their pseudocode thus looks like this:

```
if ( event_is_3D ) :
      do_3D_processing()
else : # NOT event_is_3D
      original_2D_code()
```

The representation obtains the 3D coordinates by querying the appropriate vtk3DInputDevice instance.

By keeping the original communication protocol between the representation and event handling part in place, only augmented by the Boolean flags, the modifications to the event handling part are trivial. All that is needed is to subscribe to the relevant 3D events, indicate that the current event is in fact a 3D event, and call the original widget's corresponding event handling code. For example, the pseudocode for handling 3D device motion looks like this:

```
function HandleMotion3D() :
      representation.EventIs3D()
      HandleMotion2D()
      representation.EventIsNot3D()
```

The original 2D code performs all the queries, state changes and other functions as usual. The representation however now knows whether to use 2D or 3D coordinates, and computes the interaction state or performs the interaction accordingly.

One additional task performed by the modified event handling part is changing the appearance of the virtual input device representation (i.e., the 3D pointer); this uses the mechanism with which the widgets normally change the mouse cursor, for example when the mouse is currently over a handle of the widget. The Boolean flag indicating a 3D event determines whether it is the mouse cursor or the virtual device representation that should change appearance. This of course only has an effect if the 3D pointer supports appearance changes; this is currently only supported when the representation is a vtk3DInputDeviceRepresentation, not for other representations.

Bibliography

[1] ER Abraham, "The fractal branching of an arborescent sponge," *Mar Biol*, vol. 138, pp. 503-510, 2001.

[2] R Adams and L Bischof, "Seeded region growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641-647, 1994.

[3] C Ahlberg and E Wistrand, "IVEE: an information visualization and exploration environment," in *Proceedings INFOVIS '95*, 1995, p. 66.

[4] P. G. Batchelor et al., "Quantification of the shape of fiber tracts," *Magnetic Resonance in Medicine*, vol. 55, no. 4, pp. 894-903, 2006.

[5] RA Becker and WS Cleveland, "Brushing scatterplots," *Technometrics*, vol. 29, no. 2, pp. 127-142, 1987.

[6] M Bertalmio, G Sapiro, V Caselles, and C Ballester, "Image inpainting," in *Proceedings SIGGRAPH 2000*, 2000, pp. 417-424.

[7] EA Bier, MC Stone, K Pier, W Buxton, and TD DeRose, "Toolglass and magic lenses: the see-through interface," in *Proceedings SIGGRAPH '93*, 1993, pp. 73-80.

[8] J Bloomenthal, "Calculation of reference frames along a space curve," in *Graphics Gems*, A Glassner, Ed. Boston: Academic Press, 1990, pp. 567-571.

[9] Harry Blum, "A Transformation for Extracting New Descriptors of Shape," *Models for the Perception of Speech and Visual Form*, pp. 362-380, 1967.

[10] Gunilla Borgefors, "Distance Transformations in Digital Images," *Computer Vision, Graphics, and Image Processing*, vol. 34, no. 3, pp. 344-371, 1986.

[11] CP Botha and FH Post, "Hybrid scheduling in the DeVIDE dataflow visualisation environment," in *Proceedings SimVis 2008*, Magdeburg, 2008, pp. 309-322.

[12] LD Brown and H Hua, "Magic lenses for augmented virtual environments," *IEEE Computer Graphics and Applications*, vol. 26, no. 4, pp. 64-73, 2006.

[13] J Bythell, P Pan, and J Lee, "Three-dimensional morphometric measurements of reef corals using underwater photogrammetry techniques," *Coral Reefs*, vol. 20, no. 3, pp. 193-199, Nov. 2001.

[14] K Cedilnik and P Rheingans, "Procedural annotation of uncertain information," in *Proceedings IEEE Visualization 2000*, 2000, pp. 77-83.

[15] S Cocito, S Sgorbini, A Peirano, and M Valle, "3-D reconstruction of biological objects using underwater video technique and image processing.," *J Exp Mar Biol Ecol*, vol. 279, pp. 57-70, 2003.

[16] LD Cohen and R Kimmel, "Global minimum for active contour models: a minimal path approach ," *International Journal of Computer Vision*, vol. 24, no. 1, pp. 57-78, 1997.

[17] BD Conner et al., "Three-dimensional widgets," in *Proceedings SI3D '92*, 1992, pp. 183-188.

[18] Nadine Couture, Guillaume Rivière, and Patrick Reuter, "GeoTUI: a tangible user interface for geoscience," in *Proceedings TEI '08*, Bonn, 2008, pp. 89-96.

[19] CJ Curtis, SE Anderson, JE Seims, KW Fleischer, and DH Salesin, "Computer-generated watercolor," in *Proceedings SIGGRAPH '97*, 1997, pp. 421-430.

[20] J Davis, SR Marschner, M Garr, and M Levoy, "Filling holes in complex surfaces using volumetric diffusion," in *Proceedings 3DPVT '02*, 2002, pp. 428-439.

[21] Jamshid Dehmeshki, Xujiong Ye, XinYu Lin, Manlio Valdivieso, and Hamdan Amin, "Automated detection of lung nodules in CT images using shape-based genetic algorithm," *Computerized Medical Imaging and Graphics*, vol. 31, pp. 408--417, Sep. 2007.

[22] T Deschamps, "Curve and Shape Extraction with Minimal Path and Level-Sets techniques - Applications to 3D Medical Imaging," Université Paris - IX Dauphine, Paris, PhD Thesis 2001.

[23] S Djurcilov, K Kim, P Lermusiaux, and A Pang, "Visualizing scalar volumetric data with uncertainty," *Computer & Graphics*, vol. 26, pp. 239-248, April 2002.

[24] JS Duncan and N Ayache, "Medical image analysis: progress over two decades and the challenges ahead," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 85-106, 2000.

[25] George W Fitzmaurice, Hiroshi Ishii, and William A.S. Buxton, "Bricks: laying the foundations for graspable user interfaces," in *Proceedings CHI '95*, 1995, pp. 442-449.

[26] Alexandre Gillet, Michel Sanner, Daniel Stoffler, David Goodsell, and Arthur Olson, "Augmented Reality with Tangible Auto-Fabricated Models for Molecular Biology Applications," in *Proceedings Visualization 2004*, 2004, pp. 235-242.

[27] A Gooch, B Gooch, P Shirley, and E Cohen, "A non-photorealistic lighting model for automatic technical illustration," in *Proceedings SIGGRAPH '98*, 1998, pp. 447-452.

[28] DL Gresh, BE Rogowitz, RL Winslow, and DF Scollan, "Weave: a system for visually linking 3-D and statistical visualizations, applied to cardiac simulation and measurement data," in *Proceedings IEEE Visualization 2000*, 2000, pp. 489-492.

[29] G Grigoryan and P Rheingans, "Point-based probabilistic surfaces to show surface uncertainty," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 5, pp. 564-573, 2004.

[30] G Grigoryan and P Rheingans, "Probabilistic surfaces: Point based primitives to show surface uncertainty," in *Proceedings IEEE Visualization 2002*, 2002.

[31] M Haller and D Sperl, "Real-time painterly rendering for MR applications," in *Proceedings GRAPHITE '04*, 2004, pp. 30-38.

[32] Ken Hinckley, Randy Pausch, John C Goble, and Neal F Kassell, "Passive real-world interface props for neurosurgical visualization," in *Proceedings CHI '94*, 1994, pp. 452-458.

[33] RE Horton, "Erosional development of streams and their drainage basins: hydrophysical approach to quantitative morphology," *Geol Soc Am Bull*, vol. 56, pp. 275-370, 1945.

[34] L Ibanez and W Schroeder, *The ITK Software Guide*. Clifton Park: Kitware, Inc., 2003.

[35] Hiroshi Ishii and Brygg Ullmer, "Tangible bits: towards seamless interfaces between people, bits and atoms," in *Proceedings CHI '97*, 1997, pp. 234-241.

[36] R Jesse and T Isenberg, "Use of hybrid rendering styles for presentations," in *Poster Proceedings WSCG 2003*, 2003.

[37] R Jesse, T Isenberg, B Nettelbeck, and T Strothotte, "Dynamics by Hybrid Combination of Photorealistic and Non-Photorealistic Rendering Styles," Otto-von-Guericke University, Magdeburg, Tech. Rep. 5 2004.

[38] C Johnson and AR Sanderson, "A next step: Visualizing errors and uncertainty," *IEEE Computer Graphics and Applications*, vol. 23, no. 5, pp. 6-10, 2003.

[39] PP Jonker and AM Vossepoel, "Mathematical morphology in 3D images: comparing 2D & 3D skeletonization algorithms," in *Proceedings BENEFIT Summer School on Morphological Image and Signal Processing*, Zakopane, 1995, pp. 83-108.

[40] JA Kaandorp, "Morphological analysis of growth forms of branching marine sessile organisms along environmental gradients," *Mar. Biol.*, vol. 134, pp. 295-306, 1999.

[41] JA Kaandorp and RA Garcia Leiva, "Morphological analysis of two- and three-dimensional images of branching sponges and corals," in *Morphometrics and their applications in Paleontology and Biology*, AMT Elewa, Ed. Berlin: Springer-Verlag, 2004, pp. 83-94.

[42] JA Kaandorp and J Kübler, *The algorithmic beauty of seaweeds, sponges and corals*. Heidelberg: Springer-Verlag, 2001.

[43] JA Kaandorp et al., "Morphogenesis of the branching reef coral Madracis mirabilis," *Proc. Roy. Soc. Lond. B*, vol. 272, pp. 127-133, 2005.

[44] DA Keim, "Information visualization and visual data mining," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 1-8, 2002.

[45] Kitware, Inc., *The Visualization Toolkit User's Guide.*, 2003.

[46] Arjan J Kok and Robert van Liere, "Co-location and tactile feedback for 2D widget manipulation," in *Proceedings IEEE Virtual Reality 2004*, 2004, pp. 233-234.

[47] L Lam, SW Lee, and CY Suen, "Thinning methodologies - a comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 9, pp. 869-885, 1992.

[48] Hyunjun Lee, Sungtae Kwon, and Seungyong Lee, "Real-time pencil rendering," in *Proceedings NPAR '06*, 2006, pp. 37-45.

[49] JE Lloyd and KP Dinesh, "Interactive Exploration of Remote Objects Using a Haptic-VR Interface," in *Experimental Robotics VIII.*, 0 2003, pp. 560-569.

[50] A Majumder and M Gopi, "Hardware accelerated real time charcoal rendering," in *Proceedings NPAR '02*, 2002, pp. 59-66.

[51] HB Mann and DR Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50-60, 1947.

[52] R Merks, A Hoekstra, J Kaandorp, and P Sloot, "Polyp oriented modelling of coral growth," *J Theor Biol*, vol. 228, no. 4, pp. 559-576, 2004.

[53] J Mulder, "Occlusion in mirror-based co-located augmented reality systems.," *Presence*, vol. 15, no. 1, pp. 93-107, 2006.

[54] M Näf, G Székely, R Kikinis, M E Shenton, and O Kübler, "3D Voronoi skeletons and their usage for the characterization and recognition of 3D organ shape," *Comput. Vis. Image Underst.*, vol. 66, no. 2, pp. 147-161, 1997.

[55] National Ocenic and Atmospheric Administration - National Ocean Service. (2006, January) Wikimedia Commons. [Online]. http://commons.wikimedia.org/wiki/File:Coral_polyp.jpg

[56] CW Niblack, PB Gibbons, and DW Capson, "Generating Skeletons And Centerlines From The Distance Transform," *Graphical Models and Image Processing*, vol. 54, pp. 420-437, 1992.

[57] M Nienhaus and J Döllner, "Sketchy drawings," in *Proceedings AFRIGRAPH '04*, 2004, pp. 73-81.

[58] J Northrup and L Markosian, "Artistic silhouettes: a hybrid approach," in *Proceedings NPAR 2000*, 2000, pp. 31-37.

[59] Michaël Ortega and Sabine Coquillart, "Prop-Based Haptic Interaction with Co-location and Immersion: an Automotive Application," in *Proceedings HAVE 2005*, 2005.

[60] N Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62-66, 1979.

[61] Kálmán Palágyi and Attila Kuba, "A 3D 6-subiteration thinning algorithm for extracting medial lines," *Pattern Recognition Letters*, vol. 19, pp. 613-627, May 1998.

[62] AT Pang, CM Wittenbrink, and SK Lodha, "Approaches to uncertainty visualization," *The Visual Computer*, vol. 13, no. 8, pp. 370-390, 1997.

[63] P Perona and J Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. PAMI*, vol. 12, no. 7, pp. 629-639, 1990.

[64] Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein, "Real-time hatching," in *Proceedings SIGGRAPH 2001*, 2001, p. 581.

[65] F Reinders, M E Jacobson, and F H Post, "Skeleton Graph Generation for Feature Shape Description," in *Proceedings Data Visualization 2000*, Amsterdam, 2000, pp. 73-82.

[66] P Rheingans and S Joshi, "Visualization of molecules with positional uncertainty," in *Proceedings Data Visualization 1999*, 1999, pp. 299-306.

[67] JC Roberts and MAE Wright, "Towards ubiquitous brushing for information visualization," in *Proceedings IV '06*, 2006, pp. 151-156.

[68] S Rusinkiewicz and M Levoy, "Efficient variants of the ICP algorithm," in *Proceedings 3DIM '01*, 2001, pp. 145-152.

[69] T Saito and T Takahashi, "Comprehensible rendering of 3-D shapes," *SIGGRAPH Computer Graphics*, vol. 24, no. 4, pp. 197-206, 1990.

[70] JA Sanchez and HR Lasker, "Patterns of morphological integration in marine modular organisms: supra-module organization in branching octocoral colonies," *Proc R Soc Lond B*, vol. 270, pp. 2039-2044, 2003.

[71] WJ Schroeder, KM Martin, and WE Lorensen, "The design and implementation of an object-oriented toolkit for 3D graphics and visualization," in *Proceedings IEEE Visualization '96*, 1996, pp. 93-100.

[72] JA Sethian, *Level Set Methods and Fast Marching Methods*.: Cambridge University Press, 1999.

[73] L Shaish, A Abelson, and B Rinkevich, "Branch to colony trajectory in a modular organism: pattern formation in the Indopacific coral Stylophora pistillata," *Dev Dynam*, vol. 235, pp. 2111-2121, 2006.

[74] M Sonka, V Hlavac, and R Boyle, *Image Processing, Analysis and Machine Vision*, 3rd ed.: CL-Engineering, 2007.

[75] MC Sousa, A Gooch, and B Gooch, "Illustrative scientific visualization framework," in *Proceedings CompAesth 2005*, 2005, pp. 201-208.

[76] AN Strahler, "Quantitative analysis of watershed geomorphology," *Eos Trans AGU*, vol. 38, pp. 913-920, 1957.

[77] T Strothotte, M Mausch, and T Isenberg, "Visualizing knowledge about virtual reconstructions of ancient architecture," in *Proceedings Computer Graphics International 1999*, 1999, pp. 36-43.

[78] G Taubin, "Estimating the tensor of curvature of a surface from a polyhedral approximation," in *Proceedings ICCV 1995*, 1995, pp. 902-907.

[79] RM Taylor et al., "VRPN: a device-independent, network-transparent VR peripheral system," in *Proceedings VRST '01*, 2001, pp. 55-61.

[80] J Thomson, E Hetzler, A MacEachren, M Gahegan, and M Pavel, "A typology for visualizing uncertainty," in *Proceedings Visualization and Data Analysis 2005*, 2005, pp. 146-157.

[81] Nhon Trinh, Jonathan Lester, Braden Fleming, Glenn Tung, and Benjamin Kimia, "Accurate Measurement of Cartilage Morphology Using a 3D Laser Scanner," in *Computer Vision Approaches to Medical Image Analysis*., 0 2006, pp. 37-48.

[82] J Viega, MJ Conway, G Williams, and R Pausch, "3D magic lenses," in *Proceedings UIST '96*, 1996, pp. 51-58.

[83] L Vincent and P Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583-598, 1991.

[84] C Ware and J Rose, "Rotating virtual objects with real handles," *ACM Transactions on Computer-Human Interaction*, vol. 6, no. 2, pp. 162-180, 1999.

[85] M Webb, E Praun, A Finkelstein, and H Hoppe, "Fine tone control in hardware hatching," in *Proceedings NPAR '02*, 2002, pp. 55-58.

[86] CM Wittenbrink and AT, Lodha, SK Pang, "Glyphs for visualizing uncertainty in vector fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 3, pp. 266-279, 1996.

[87] Wenjie Xie, Robert P Thompson, and Renato Perucchio, "A topology-preserving parallel 3D thinning algorithm for extracting the curve skeleton," *Pattern Recognition*, vol. 36, pp. 1529-1544, July 2003.

[88] I Yekutieli and BB Mandelbrot, "Horton-Strahler ordering of random binary trees," *Journal of Physic A: Mathematical and General*, vol. 27, no. 2, pp. 285-293, 1994.

[89] Brett M Young et al., "Polyp Measurement with CT Colonography: Multiple-Reader, Multiple-Workstation Comparison," *Am. J. Roentgenol.*, vol. 188, pp. 122--129, Jan. 2007.

[90] G Zachmann, "Distortion correction of magnetic fields for position tracking," in *Proceedings CGI '97*, 1997, pp. 213-220.

[91] T Zuk and S Carpendale, "Theoretical analysis of uncertainty visualizations," in *Proceedings Visualization and Data Analysis 2006*, 2006, pp. 66-79.

List of Figures

Figure 1.1: Global overview of the measurement pipeline. 1
Figure 1.2: A coral polyp. 2
Figure 1.3: Photographs of the three coral specimens. 3
Figure 1.4: Skeletons of the corals in Figure 1.3. 4
Figure 1.5: The model of a coral colony. 5

Figure 3.1: Steps from CT scan to skeleton. 15
Figure 3.2: Volume rendering of coral with noise. 16
Figure 3.3: Background noise measurement for each of the three objects. 17
Figure 3.4: Additional background noise measurement. 17
Figure 3.5: Segmentation of the three specimens. 20
Figure 3.6: Difficulties in coral images. 23
Figure 3.7: The polygonal model of the test object. 30
Figure 3.8: Graphs of each measurement. 32
Figure 3.9: Error histograms for each input object. 34
Figure 3.10: The relative error of the Xie algorithm. 34
Figure 3.11: Sample permutations and measurements. 35

Figure 4.1: An example skeleton. 39
Figure 4.2: Maximum thickness *da* (*a*-sphere). 42
Figure 4.3: Minimum thickness *db* (*b*-sphere). 42
Figure 4.4: Terminal thickness *dc*. 43
Figure 4.5: Branching rate shown in red. 43
Figure 4.6: Branch spacing. 43
Figure 4.7: Geotropy angles. 44
Figure 4.8: Branching angles. 44
Figure 4.9: A 3D visualization of the measured *a*-sphere diameters. 48
Figure 4.10: Histograms for branches of order 1 (terminal branches). 50
Figure 4.11: Histograms for branches of order 2. 50
Figure 4.12: Histograms for branches of order 3. 51
Figure 4.13: Histograms for branches of order 4. 51
Figure 4.14: Histograms for the measurements. 52

Figure 5.1: Highlighted loops in the skeleton. 56
Figure 5.2: An overview of the exploration system. 57
Figure 5.3: Selection of measurements using a histogram. 58

Figure 5.4: Close-up of tube worms 61
Figure 5.5: Close-up of a skeleton with spurious branches. 61
Figure 5.6: Brushing of outliers. 62
Figure 5.7: A particularly bushy skeleton. 62
Figure 5.8: Various rendering styles. 65
Figure 5.9: A line with varying uncertainty... 67
Figure 5.10: Mapping of uncertainty to photorealism. 68
Figure 5.11: A combined visualization of uncertainty... 69
Figure 5.12: A close-up of the coral surface visualization with uncertainty. 69
Figure 5.13: Direct rendering of uncertainty to grayscale. 70
Figure 5.14: Skeleton graph visualization with uncertainty.. 72
Figure 5.15: Skeleton data set with uncertainty visualized using the tube approach........... 73

Figure 6.1: The basic model refinement process. 79
Figure 6.2: Refinement process with tangible props.. 79
Figure 6.3: The test object with a sphere attached to it... 82
Figure 6.4: Diagrams of the four methods that were compared... 82
Figure 6.5: Screen capture of the tangible Magic Lens. 85
Figure 6.6: Graphs of the measurement results... 86
Figure 6.7: Interaction through collision detection... 89
Figure 6.8: Obtaining the corrected model. 89
Figure 6.9: An overview of the transformations and tolerances. 94
Figure 6.10: Improper occlusion effects in augmented reality using a video feed............... 99
Figure 6.11: Tangible props printed from coral data... 101
Figure 6.12: Tangible coral prop with attached sensor and stylus. 101
Figure 6.13: Interactive coral measurement application.. 103

Figure A.1: The processing of events. 112

Interactive Measurements of Three-Dimensional Branching Objects

Summary

One of the methods to learn more about the shape of an object is to measure it. Measuring the object yields quantitative data which can subsequently be used to gain new knowledge about the object. When the same metrics are applied to a number of objects, it becomes possible to perform an objective quantitative comparison of these objects. Performing the measurements using an interactive measurement system makes it possible to obtain and analyze large amounts of accurate measurements. This makes a more thorough analysis possible.

Branching objects are ubiquitous in nature. Examples include trees, vascular systems, neurons, bronchial tissue trees, snowflakes, river networks, sea sponges, and branching corals. The focus of this thesis is on measuring the shape of marine branching corals. While these marine creatures are relatively simple and small organisms, they form large underwater colonies with a wide variety of different shapes that are in part dependent on environmental factors. These colonies are the cornerstone of many marine environments. Accurate quantitative analysis of these shapes enables biologists to better understand the mechanisms which govern coral growth.

In this thesis, marine branching corals are subjected to quantitative morphological analysis. For the first time, a fully three-dimensional analysis is performed on 3D Computed Tomography (CT) scans of coral specimens. From these images the 3D branching structure is extracted by computing the morphological skeleton; this is an intermediary representation that is used to locate the relevant features of the coral. The extraction is not without difficulties, as different extraction algorithms have different sensitivity to noise and various other artifacts that are present in the data. This is solved in part by quantitatively comparing different algorithms and selecting the most appropriate one, and in part through interactive editing of the data to remove some of the artifacts.

The extracted skeleton is used to perform the actual measurements. Some of the metrics use only the skeleton, while others use the skeleton to perform measurements on the image, for example to measure the thickness. Three coral specimens were measured and compared in this manner; the results were consistent with previously observed trends for the species, which validates the methodology.

The use of visualization and interaction for a number of purposes is researched. The coral is explored using interactive 3D visualizations. In addition, such visualizations are used to edit the data in order to remove artifacts which would affect the analysis. The results of the measurements are also explored in a highly interactive manner. Interactive selection in linked 2D statistic data plots and 3D visualizations is used, where selection in

one view is restricts or highlights corresponding parts of the other view. Finally, the use of non-photorealistic rendering techniques is used to add uncertainty to 3D visualizations.

Interacting with complex objects such as corals can be difficult and confusing to the user, as only visual information is relied upon. A more natural user interface also makes use of the sense of touch; this is achieved by using tangible input props and augmented reality. These props are created by using 3D printing technology to produce a tangible representation of the data. Instead of using traditional input devices such as a mouse or 3D pen, the user interacts with the coral by holding it in their hand and manipulating it just like a real object. Augmented reality techniques are then used to annotate this hand-held prop with relevant visualizations, enabling efficient interaction with the real-world object.

The methods and techniques in this thesis allow coral researchers to perform accurate measurements on their specimens. However, most of the concepts and techniques described in this thesis are not limited to coral. Skeletons are used to quantify and analyze a wide variety of objects. Interactive visualizations are also applicable to any field. Finally, printed tangible props can be created to interact with any type of data.

Interactive Measurements of Three-Dimensional Branching Objects

Samenvatting

Een van de methoden om meer over de vorm van een object te weten te komen is het te meten. Het meten van het object resulteert in kwantitatieve data welke vervolgens gebruikt kan worden om nieuwe kennis over dit object te verwerven. Als dezelfde metrieken worden toegepast op een aantal objecten wordt het mogelijk om een objectieve kwantitatieve vergelijking tussen deze objecten te maken. Door de metingen te verrichten middels een interactief meetsysteem wordt het mogelijk om grote hoeveelheden nauwkeurige metingen te verkrijgen en te analyseren. Hierdoor wordt een grondigere analyse mogelijk.

Vertakkende objecten zijn alomtegenwoordig in de natuur. Voorbeelden zijn onder meer bomen, vaatstelsels, neuronen, longweefsel-bomen, sneeuwvlokken, riviernetwerken, zeesponzen, en vertakkende koralen. De focus van dit proefschrift ligt op het meten van de vorm van vertakkende zeekoralen. Hoewel deze zeedieren relatief eenvoudige en kleine organismen zijn, vormen ze onder water grote kolonies met een breed scala aan verschillende vormen, die deels afhankelijk zijn van omgevingsfactoren. Deze kolonies vormen de hoeksteen van vele zeemilieus. Nauwkeurige kwantitatieve analyse van deze vormen geeft biologen een beter inzicht in de mechanismen die de groei van koraal regelen.

In dit proefschrift worden vertakkende zeekoralen onderworpen aan een kwantitatieve morfologische analyse. Voor het eerst is een volledig drie-dimensionale analyse uitgevoerd op 3D Computed Tomography (CT)-scans van koraalspecimens. Uit deze beelden word the 3D vertakkingsstructuur geëxtraheerd door het morfologisch skelet te berekenen; dit is een intermediaire representatie die gebruikt wordt om de relevante kenmerken van het koraal te localiseren. Deze extractie is niet zonder problemen, aangezien verschillende extractiealgoritmes verschillen in hun gevoeligheid voor ruis en andere artefacten die in de data aanwezig zijn. Dit wordt opgelost deels door verschillende algoritmes kwantitatief te vergelijken en het meest toepasselijke algoritme te selecteren, en deels door interactieve bewerking van de data om sommige artefacten te verwijderen.

Het geëxtraheerde skelet wordt gebruikt om de daadwerkelijke metingen uit te voeren. Sommige metingen gebruiken alleen het skelet, terwijl andere het skelet gebruiken om metingen te verrichten op de beelden, bijvoorbeeld het meten van de dikte. Drie koraalspecimens zijn op deze manier gemeten en vergeleken; de resultaten kwamen overeen met eerder geconstateerde trends voor deze soort, wat aantoont dat de methodologie correct is.

Het gebruik van visualisatie en interactie voor een aantal doeleinden is onderzocht. Het koraal wordt nauwkeurig onderzocht met behulp van interactieve 3D visualisaties. Bovendien worden dergelijke visualisaties ook gebruikt voor het bewerken van de gegevens, om artefacten te verwijderen die de analyse zouden beïnvloeden. De resultaten van de metingen worden ook geanalyseerd op een zeer interactieve manier. Er wrodt gebruik gemaakt van interactieve selectie in aan elkaar gekoppelde 2D statistische plots en 3D visualisaties, waarbij selectie in de ene visualisatie wordt gebruikt om de overeenkomstige delen van de andere visualisatie te beperken of te markeren. Tenslotte worden niet-fotorealistische rendertechnieken gebruikt om de onzekerheid toe te tonen in 3D-visualisaties.

Interactie met complexe objecten zoals koralen kan lastig en verwarrend zijn voor de gebruiker, aangezien alleen op visuele informatie wordt vertrouwd. Een meer natuurlijke gebruikersinterface maakt ook gebruik van de tastzin, dit wordt bereikt door gebruik te maken van tastbare input props en augmented reality. De props worden gemaakt met behulp van 3D-printing technologie om een tastbare representatie van de gegevens te produceren. In plaats het gebruik van traditionele invoerapparaten zoals een muis of 3D-pen, interacteert de gebruiker met het koraal door het in de hand te houden en het te hanteren net als een elk normaal voorwerp. Vervolgens wordt gebruik gemaakt van augmented reality-technieken om deze prop te voorzien van relevante visualisaties, waardoor efficiënte interactie met dit echte voorwerp mogelijk wordt.

De methoden en technieken in dit proefschrift laten koraalonderzoekers nauwkeurige metingen uitvoeren op hun specimens. Toch zijn de meeste van de in dit proefschrift beschreven concepten en technieken niet beperkt tot koraal. Skeletten worden gebruikt voor het kwantificeren en analyseren een breed scala aan objecten. Interactieve visualisaties vinden ook toepassing op allerlei gebieden. Ten slotte kunnen geprinte tastbare representaties gemaakt worden voor interactie met elk soort data.

Interactive Measurements of Three-Dimensional Branching Objects

Streszczenie

Jednym ze sposobów, aby dowiedzieć się więcej na temat kształtu przedmiotu jest zmierzenie go. Pomiar przedmiotu tworzy dane kwantytatywne (numeryczne), które mogą być następnie wykorzystywane w celu uzyskania nowej wiedzy o przedmiocie. Gdy te same pomiary są stosowane do kilku przedmiotów, staje się możliwe, aby wykonać obiektywne kwantytatywne porównanie tych przedmiotów. Wykonywanie pomiarów za pomocą interaktywnego systemu do mierzenia pozwala na uzyskanie i analizowanie dużych ilości dokładnych pomiarów, co umożliwia bardziej szczegółową analizę.

Rozgałęzione przedmioty są wszechobecne w przyrodzie. Przykłady to drzewa, układy naczyniowe, neurony, drzewa tkanki oskrzelowej, płatki śniegu, sieci rzek, gąbki morskie, i rozgałęzione koralowce. Temat tej pracy to pomiar kształtu morskich rozgałęzionych koralowców. Te morskie stworzenia są stosunkowo prostymi i małymi organizmami, jednak tworzą one wielkie podwodne kolonie o bardzo różnorodnych kształtach, które są częściowo uzależnione od czynników środowiskowych. Te kolonie są podstawą wielu środowisk morskich. Dokładna analiza kwantytatywna tych kształtów umożliwia biologom lepsze zrozumienie mechanizmów które regulują wzrost koralowców.

W tej pracy, morskie rozgałęzione koralowce są poddane kwantytatywnej analizie morfologicznej. Po raz pierwszy, w pełni trójwymiarowe analizy są wykonane na obrazach koralowców z tomografii komputerowej (CT). Z tych obrazów trójwymiarowa struktura rozgałęzień jest ekstrahowana – ustalona poprzez wyliczenie morfologicznych szkieletów; jest to pośrednia reprezentacja obrazów, która jest używana w celu zlokalizowania odpowiednich cech koralowców. Ekstrakcja nie jest bez trudności: różne algorytmy do ekstrakcji mają rozbieżną wrażliwość na szum i inne artefakty, które znajdują się w danych. Ten problem został rozwiązany częściowo przez kwantytatywne porównanie różnych algorytmów i wybranie najbardziej stosownego, a częściowo poprzez interaktywną modyfikację danych w celu usunięcia niektórych artefaktów.

Wyliczony szkielet jest wykorzystywany do wykonania zasadniczych pomiarów. Niektóre pomiary dotyczą tylko szkieletu, a inne używają szkielet, aby wykonać pomiary na obrazie, na przykład pomiar grubości. Trzy okazy koralowców zostały zmierzone i porównane w ten sposób; wyniki były zgodne z poprzednio zaobserwowanymi trendami dla tego gatunku, co potwierdza zgodność metodologii.

Zbadane zostało także zastosowanie wizualizacji i interakcji do kilku celów. Koralowce zostały zbadane za pomocą interaktywnej wizualizacji w 3D. Ponadto takie wizualizacje były używane do poprawiania danych w celu usuwania artefaktów, które mogłyby wpłynąć na analizę. Wyniki pomiarów zostały zbadane w sposób wysoce interaktywny. Interaktywny wybór w połączonych statystycznych wykresach w 2D i wizualizacjach w 3D

został zastosowany w taki sposób, że wybranie części danych w jednym widoku ograniczało albo podkreślało powiązane części drugiego widoku. Wreszcie, artystyczne techniki rysowania zostały wykorzystywane w celu pokazania niepewności w wizualizacjach 3D.

Interakcja ze skomplikowanymi przedmiotami, takimi jak koralowce, może być trudna i myląca dla użytkownika, jeśli musi on wyłącznie polegać na informacji wzrokowej. Bardziej naturalny interfejs wykorzystuje także zmysł dotyku; jest to osiągnięte za pomocą namacalnych rekwizytów wejściowych i *augmented reality*. Te rekwizyty są stworzone przy użyciu technologii druku 3D, który produkuje namacalne reprezentacje danych. Zamiast korzystania z tradycyjnych urządzeń wejściowych, takich jak mysz lub pióro 3D, użytkownik dokonuje interakcji z koralowcem trzymając go w ręce i manipulując nim tak jak prawdziwym przedmiotem. Technika *augmented reality* jest następnie wykorzystana do dodania stosownych wizualizacji do rekwizytu trzymanego w ręce, co umożliwia skuteczną interakcję z tym fizycznym przedmiotem.

Metody i techniki opisane w tej pracy pozwalają naukowcom badającym koralowce prowadzić dokładne pomiary okazów. Jednakże większość pojęć i technik opisanych w tej pracy nie jest ograniczona do koralowców. Szkielety są wykorzystywane do kwantyfikacji i analizy różnorodnych przedmiotów. Interaktywne wizualizacje mają również zastosowanie w niemal każdej dziedzinie, a drukowane namacalne rekwizyty można utworzyć w celu interakcji z danymi wszelkiego rodzaju.

Curriculum Vitae

Krzysztof Jakub Kruszyński was born on October 4th, 1977 in Warszawa, Poland. In June 1996 he received his Gymnasium diploma at the Vossius Gymnasium in Amsterdam. The following year he started studying Computer Science, with a specialization in Computer Systems, at the Vrije Universiteit in Amsterdam. Here he received his Master of Science degree, with distinction, in August 2003. His graduation project was performed under the supervision of prof.dr.ir. Henri E. Bal and dr. Tom van der Schaaf, and it involved creating a video playback application for use on a large cluster-driven tiled display. In February 2004, he joined the Visualization and 3D User Interfaces theme of the Centrum Wiskunde & Informatica (CWI) in Amsterdam. There he performed the Ph.D. research which is described in this thesis, under the supervision of prof.dr.ir. Robert van Liere.