Patient-specific anatomical illustration via model-guided texture synthesis

Ilknur Kaynar Kabul

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2012

Approved by:

Stephen M. Pizer

Russell Taylor

Julian Rosenman

Marc Niethammer

Svetlana Lazebnik

Abstract

ILKNUR KAYNAR KABUL: Patient-specific anatomical illustration via model-guided texture synthesis.
(Under the direction of Stephen M. Pizer.)

Medical illustrations can make powerful use of textures to attractively, effectively, and efficiently visualize the appearance of the surface or cut surface of anatomic structures. It can do this by implying the anatomic structure's physical composition and clarifying its identity and 3-D shape. Current visualization methods are only capable of conveying detailed information about the orientation, internal structure, and other local properties of the anatomical objects for a typical individual, not for a particular patient. Although one can derive the shape of the individual patient's object from CT or MRI, it is important to apply these illustrative techniques to those particular shapes. In this research patient-specific anatomical illustrations are created by model-guided texture synthesis (MGTS).

Given 2D exemplar textures and model-based guidance information as input, MGTS uses exemplar-based texture synthesis techniques to create patient-specific surface and solid textures. It consists of three main components. The first component includes a novel texture metamorphosis approach for creating interpolated exemplar textures given two exemplar textures. This component uses an energy optimization scheme derived from optimal control principles that utilizes intensity and structure information in obtaining the transformation. The second component consists of creating the model-based guidance information, such as directions and layers, for that specific model. This component uses coordinates implied by discrete medial 3D anatomical models ("m-reps"). The last component accomplishes exemplar-based texture synthesis by textures whose characteristics are spatially variant on and inside the 3D models. It considers the exemplar textures from the first component and guidance information from the second component in synthesizing high-quality, high-resolution solid and surface textures.

Patient-specific illustrations with a variety of textures for different anatomical models, such as muscles and bones, are shown to be useful for our clinician to comprehend the shape of the models under radiation dose and to distinguish the models from one another.

In honor of

my parents, Mukaddes and Ihsan Kaynar,

and

my husband, Onur Kabul

Acknowledgments

I would like to express my gratitude to people who have been important to this work and to my life as a graduate student at UNC. So many people have helped and inspired me both in research and in life and it has been a privilege and honor for me to be associated with all these people.

First and foremost I am thankful to Steve Pizer, my doctoral advisor. Without his guidance and encouragement I do not think I would have been able to finish this dissertation. He taught me scientific principles in the medical image analysis and visualization fields, helped me to achieve all the doctoral milestones and inspired me with his outstanding passion and dedication in research. It has been a privilege to be one of his doctoral advisees. He has been a great mentor to me.

I am grateful to my committee members for their comments and suggestions. Their research expertise and critical thinking were extremely valuable and helped me in clarifying specific research aims at different stages of the research. Julian Rosenman has been a visionary leader and a generous patron of the "Netterly Visualization" project. He has been actively involved in my dissertation research and has been of great help in developing major ideas in this dissertation. Marc Niethammer has lent me freely his valuable knowledge about image registration for solving the texture metamorphosis in this dissertation. I want to thank Russell Taylor for his critical comments and advices as a visualization researcher. He has been equally constructive and helpful in numerous discussions regarding my dissertation research. I also want to thank Svetlana Lazebnik for her feedback and advice.

The entire MIDAG group has been supportive and helpful over those years. In particular, I want to thank Derek Merck for being a supportive and charitable partner on my dissertation work. Last but not least, my office mate during the most productive years of my dissertation, Xiaoxiao Liu , has been a very good companion at work. Positive energy in the office definitely helped me to handle dissertation writing stress.

I would like to thank my husband, Onur, and my daughter, Nilufer, for their invaluable support during this long journey. My parents, Ihsan and Mukaddes Kaynar, and my sister, Gulnur Unal, have all contributed hugely to this project by being unconditionally supportive and boundlessly patient for what I chose to accomplish.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Realistic images of medical data, such as photos of anatomical organs, may be overwhelming for non-expert viewers because they contain too much information and irrelevant detail, making it hard to process and grasp the most salient information in the scene. Illustrations play an important role in solving this problem. Medical illustrations use different artistic techniques to depict features in an expressive way, while simultaneously providing insight into the underlying data. These illustrations consider objects carefully in order to convey information and the relation between important features of interest.

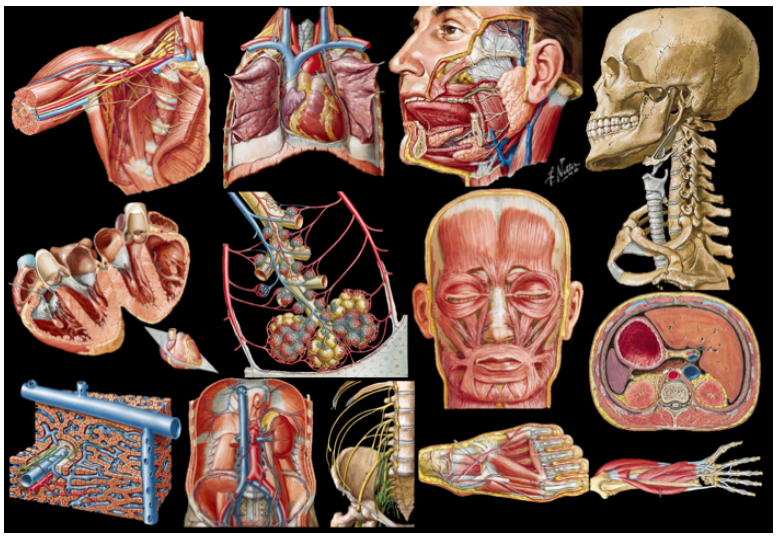

Figure 1.1: Samples from Frank H. Netter's illustrations (@2011 Elsevier).

Medical illustrations of different anatomical structures in textbooks are commonly used for many purposes, such as increasing the communication between clinicians and helping clinicians understand complex medical images. In medical illustrations textures are used as a means to distinguish structures and regions from one another by appearance, as well as to convey local structural detail, such as orientation, scale and material, and the object's 3D shape. Frank H. Netter (Netter (2009)), sometimes referred to as "Medicine's Michelangelo", used textures for identifying anatomical models and the regions inside them on a typical human (Figure 1.1). This dissertation focuses on obtaining this effect for particular patient data. Throughout this dissertation I will refer to this type of rendering as *Netterly rendering*. In this dissertation rendering the interior and exterior regions of patient-specific anatomical models is achieved using material-specific structural textures that are synthesized for the objects specific to each patient. This yields patient-specific Netterly renderings, which the user can interact with and manipulate in a 3D environment.

Many anatomical objects have specific local material properties, such as material type, scale, and orientation, over the surface of the model and inside the model. Frank Netter used textures and the variation in them as a means of conveying material properties in his illustrations. To achieve that goal, he used different texturing techniques depending on the anatomical organ:

1. Isotropic textures: Some anatomical organs do not have much variation in their material characteristics on or inside their surfaces. These organs are mostly illustrated using the same texture in all regions. For instance, organs like the thyroid and the parotid are illustrated with blobby textures which represent that these organs are glands (Figure 1.2).

2. Anisotropic textures: Many anatomical structures are composed of different materials or the same materials with different orientations and scales. These features can be illustrated by creating a texture that mimics these variations. Most of these

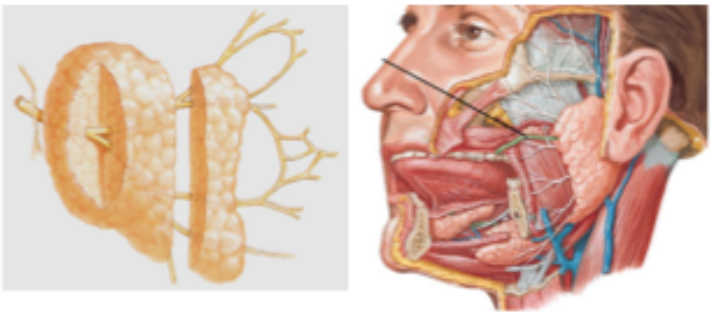

Figure 1.2: Anatomical illustrations of the parotid gland (@2011 Elsevier). (Left) The parotid texture is isotropic both inside and outside the model. (Right) The parotid illustrated inside the head and neck region with the other organs.

organs are shown with appearances that suggest features arranged along, across and around the object, and through the interior of the object. Some of these features are illustrated on the surface of the object or inside the objects, depending on that specific organ. For example, surface layers are illustrated using different tissue-specific textures for the organs in the digestive system. The layered muscle linings of the duodenum and the stomach have respective directional components along and around the object. In some of the layers they propagate along the object, whereas in the others they propagate around the object. Differently oriented surface textures are used to illustrate different layers of the model through depth for these organs (Figure 1.3).

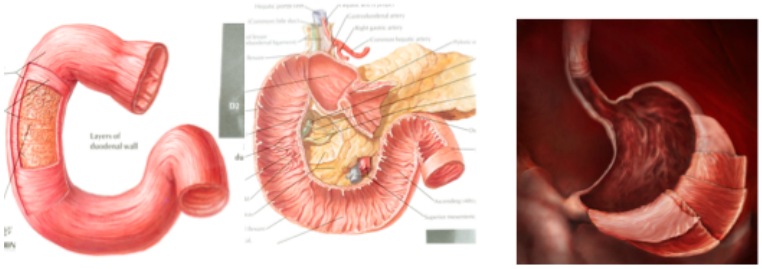

Figure 1.3: (Left and middle) Illustrations of Frank Netter for the duodenum (@2011 Elsevier); (right) anatomical illustration of the stomach.

For anatomical organs such as the sternocleidomastoid muscles (scm), volumetric

textures are used to depict the appearance of muscle tissues in these organs. These textures (Figures 1.4, 1.5, and 1.6) are oriented according to the shape of that specific model.

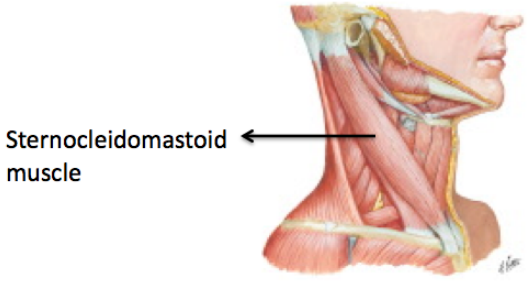

Figure 1.4: Netter illustration shows the head and neck region of a typical patient. The texture on the sternocleidomastoid muscle (scm) is oriented along the model. (@2011 Elsevier)

3. Material-specific textures: In some organ illustrations different regions in the interior of the models are depicted using tissue-specific textures. For example, the mandible has many tissue types (Figure 1.5 - right): osseous tissue, marrow, endosteum, and periosteum. Osseous tissue makes up bone, also called bone tissue, and endosteum and periosteum line the outside and inside surfaces, respectively, of the bony tissue. All of these layers are illustrated through the object with different tissue-specific textures in Netter's illustrations.

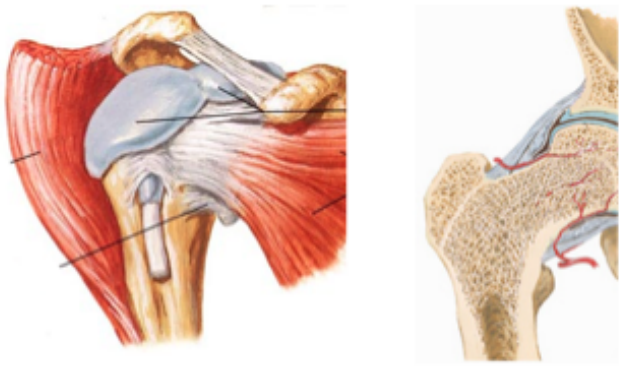

Figure 1.5: Illustrations show the change in materials for (left) the muscle, and (right) the bone. (@2011 Elsevier)

4. Object-relative textures: For some organ illustrations the material textures look different depending on the orientation of an anisotropic clipping plane. For example, the material texture on the outside of the scm looks like stream of lines, whereas when you cut the scm, you see circular blobs (Figure 1.6).

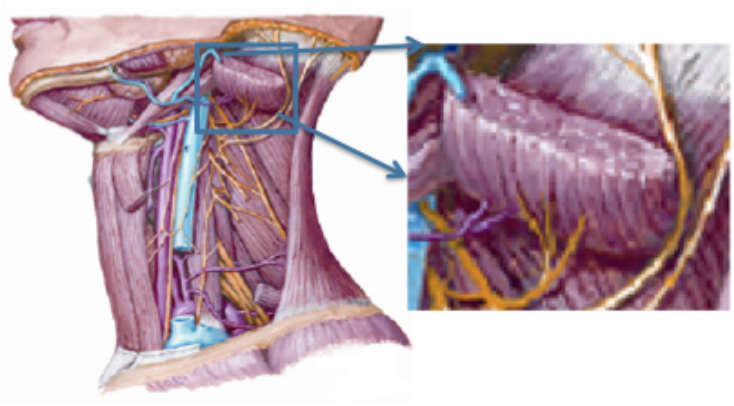

Figure 1.6: Netter illustration shows the scm with a clipping plane to visualize the interior of the organ. (@2011 Elsevier)

5. Structural textures: In some of the organ illustrations the materials are composed of other structures. These organs are mostly illustrated with textures that have structural primitives on a uniform background. For instance, the prostate is illustrated with a glandular texture on a uniform red background (Figure 1.7). In some of the illustrations the models are illustrated with more glandular primitives, and in some they are illustrated with fewer primitives.

While Netter visualized a typical patient using the approaches mentioned above, in this dissertation a novel framework called Model Guided Texture Synthesis (MGTS) is proposed for obtaining illustrative visualizations of patient-specific medical organs by creating high quality region-specific textures, both on the surface and interior of the models, using the above approaches. MGTS uses medical illustrations from Netter (2009) as guidance to select the type of texture and texture variation specific to the anatomical model. It uses regional variations in addition to orientation and scale changes as guidance in texture generation. It synthesizes textures using 2D exemplar

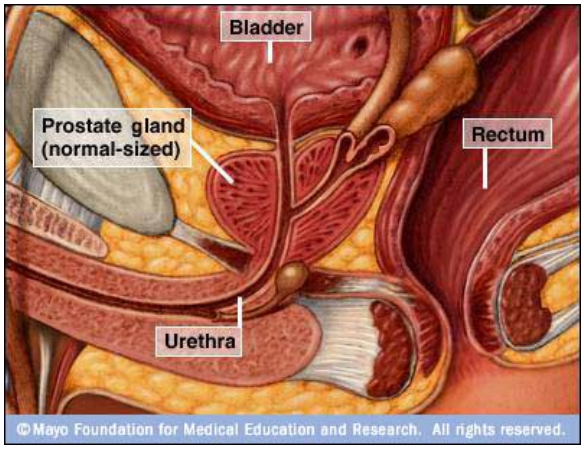

Figure 1.7: Anatomical illustration of the prostate with the other organs around it (@Mayo foundation for medical education and research).

textures using a model-specific volumetric coordinate system. The methods that are developed in MGTS are intended to make anatomical illustrations useful not as references to "normal" anatomy but rather to provide understanding of the anatomy of the particular clinical patient being examined, planned, or operated on. The main advantage of this approach compared to traditional medical illustration is the interactivity and real-time manipulation of the acquired, patient-specific data in 3D by distinguishing the organs, their shapes and the regions inside them.

1.1 Clinical Application Areas

The application of MGTS to actual patient data combines the power of traditional illustration techniques with computer-based renderings, but now it can be applied to actual patient data, instead of a canonical medical dataset. These illustrations can be used to understand the shape and material characteristics of the organs for identification purposes. To achieve these purposes in different applications, patient-specific textured organs can be integrated with other data sources by taking advantage of the multimodality renderings. For instance, the textured patient-specific anatomical models can be easily loaded into the Model Guided Rendering (MGR) framework which is presented

by Merck (2009). The MGR framework has the capability of integrating many data sources into a single environment. A user (e.g., clinician, physician) can load the acquired data set (e.g., CT, MRI) and the textured, segmented anatomical organs in a single environment. This capability makes it possible to understand the position of organs with respect to each other and with respect to the volume-rendered acquired data. S/he can interact with the organs in a single 3D environment, can cut the anatomical organs via clipping planes to see the tissues inside the organs, and can visualize the textures on the CT slices.

The integration of patient-specific textured models, obtained using MGTS, with other data sources in a single environment (e.g., the MGR framework) makes possible its use for many applications:

1. Education and training in medicine: Medical students use anatomy textbooks to understand anatomical structures. Textured depictions of the organs of a typical patient in textbooks have been shown to facilitate the learning process. However, students also need to study a collection of patients, in which the shape of the organs differs from one patient to another patient. Patient-specific data sets provide actual case studies in pathology and phenotypical variance. There is a benefit for the students to understand each patient from the collection, since these data sets are more relevant to clinical and research applications than typical data sets. Our patient-specific illustrations can help to solve this problem by creating patient-specific visualizations. Such a tool can be applied for education and training of nurses and medical students, as well as surgical residents.

2. Communication in Medicine: Medical illustrations are very important tools for communicating complex anatomical structures and their relationships (Bruckner (2006)). Showing the same detailed pictures to all types of people involved in a treatment process and in the important decisions may not be a good idea since the understanding of each person from the same picture might change depending on

their knowledge level. In addition, using only hand-drawn anatomic illustrations of a typical patient to describe and discuss a patient case might not be the best way. Thus, patient-specific illustrations can be used to guide subsequent work in an optimal direction.

(a) Doctor-patient communication: Educating patients about the diagnosis and treatment plan is very important. However, acquired patient data sets, such as MRI and CT, are hard for the patients to understand. The data should be presented to them in an understandable and effective way. Patient-specific Netterly illustrations can be used to improve the communication between patients and doctors by helping the patients to recognize the organs and to visualize the change in the shape of the organs due to a procedure or disease.

(b) Doctor-doctor communication: Geologists and geophycisists take advantage of interactive, communication-friendly interpretations of seismic data illustrations during the analysis and discussion of the acquired data sets (Patel et al. (2009)). The same principle can be applied in the medical field using patient-specific illustrative visualization of the acquired data sets. Textured models inside the MGRview framework (Merck (2009)) can be used by doctors during the analysis and planning of a patient's treatment.

3. Treatment planning and dose evaluation in 3D: In radiation treatment planning, visualization of radiation dose on 3D models is very important to convey how the dose is distributed inside the affected region. Many applications use color to display dose magnitude, making it necessary to use another channel for identification purposes. Texture can be used as an option for achieving identification and shape communication purposes. Even under radiation dose, texturing can help to retain these properties.

1.2 Illustrative Visualization

There are many techniques capable of illustrating the anatomical organs in the human body. These techniques can be categorized depending on the target application, or depending on the data that they use for illustration. While the illustrations of a typical person can be enough for teaching the anatomical structures in the body, the visualization of a specific patient data is necessary for clinical and research applications (Bruckner (2006)). This dissertation adapts the techniques used in the illustrations of a typical patient to create illustrations of real patient data sets. Thus, the techniques that are related with these fields are summarized briefly in this section:

1. Medical textbook illustrations: These visual materials are designed to communicate medical and biological knowledge. They are used effectively in all major markets, including advertising, editorial, institutional and instructional. Many medical illustrations use different stylized rendering techniques, such as silhouettes and contours, pen and ink, stippling, and hatching to obtain different depictions to clarify the structural properties (shape, size, appearance, etc.) of the organs and the spatial relations between them. The main purpose of these techniques is to emphasize different levels of information (e.g., different levels of information in the same image) in order to communicate to the viewer. They can be highly realistic and anatomically accurate, or abstract and wildly conceptual depending on the target usage. Frank H. Netter, M.D. illustrated a series of atlases for each organ system, which cover human anatomy, embryology, physiology, pathology, and pertinent clinical features of the diseases arising in each system (Netter (2009)). His illustrations capture the essential message of a given anatomical region or physiological / pathological process and visualize them using life-like coloring and rendering techniques in a beautiful and compelling way. These illustrations are widely used to inform and educate both medical students and doctors. Netter

used different techniques, such as clipping, contours, and textures, as a means of conveying information about the anatomical models. This thesis applies to the texturing aspect of his illustrations.

2. Computer-supported medical visualizations: These visualizations create interactive and effective representations of patient-specific data in order to create understanding of the information present in the given data. Such visualizations are widely used in many different areas such as clinical research and medical education. Current medical visualization techniques typically use untextured volume rendering and surface rendering approaches (Levoy (1988)). However, these visualization methods have weaknesses in conveying detailed information about the orientation, internal structure, and other local properties of the anatomical objects for a particular patient because imaging modalities such as CT or MRI do not capture this information. For example, Figure 1.8 shows an illustration of a head and neck region and a corresponding volume-rendering of the CT data. In this figure the textures help to visualize details of the organs.

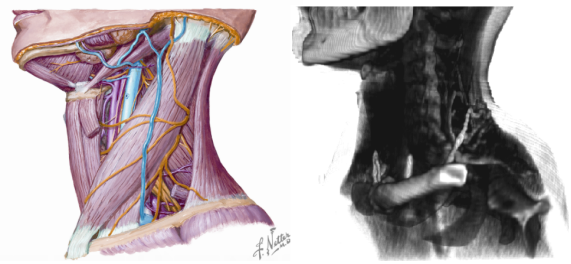

Figure 1.8: A scientific illustration (@2011 Elsevier) (left) shows more general information than the corresponding volume-rendered CT data (right).

3. Illustrative visualization: This approach has become very popular in the last decade for obtaining interactive and expressive visualizations of medical data using abstraction (Bruckner (2006), Bruckner et al. (2010), Viola et al. (2010), Burns et al. (2007)). As in traditional illustrations, the main purpose is to convey complex structures or procedures in an easily understandable way using different degrees of

abstraction for different purposes. Compared to the realistic images of anatomical models, these illustrations convey more information in an attractive, efficient, and understandable way. Different abstraction techniques consider which parts of the scene are going to be rendered and how they are going to be rendered depending on the purpose of application. This dissertation focuses on how different objects can be rendered to differentiate structures and regions in anatomical organs.

1.3 Overview of Approach

MGTS synthesizes progressively changing solid or surface textures for patient-specific anatomical models by considering the orientation and variation of the textures inside the model. The input to MGTS consists of a 3D triangular mesh model, a model-based coordinate system for the model (in our method obtained from a medial model of the object), and a predetermined set of 2D exemplar textures. The output is a textured model. MGTS is a post-segmentation process, and it synthesizes model-specific surface and solid textures for the segmented anatomical models using exemplar-based texture synthesis techniques.

MGTS consists of three main components. The first two components create the inputs for the texture synthesis process. The first component includes a novel texture metamorphosis approach for creating interpolated exemplar textures given two exemplar textures. This component uses an energy optimization scheme derived from optimal control principles that utilizes intensity and structure information in obtaining the transformation. The second component creates model-based guidance information, such as directions and layers, for that specific model. This component uses coordinates implied by discrete medial 3D anatomical models ("m-reps"). M-reps nicely provide a 3D along-object, across-object, and through-object parameterization on and within the model. In the MGTS framework, two important features in texture synthesis step are considered:

the guidance vector field and the variation of the textures inside the model. Both of them are obtained from the model-based coordinate systems.

The last component accomplishes exemplar-based texture synthesis of progressively changing textures on and inside the 3D models. It considers the exemplar textures from the first component and guidance information from the second component in synthesizing high-quality, high-resolution solid and surface textures. The decision of what kind of textures is going to be generated, solid or surface, is done based on the illustrations in the anatomical textbooks. In some case it is useful to understand the texture within the object, in which texturing on a cutting plane is effective. The appearance of the cutting plane texture depends on the plane's orientation (through or along the object). Moreover, the number and shape of the structural primitives (textons) are taken into account in creating solid textures for some of the organs, such as the prostate. Figure 1.9 shows the pipeline of the approach.

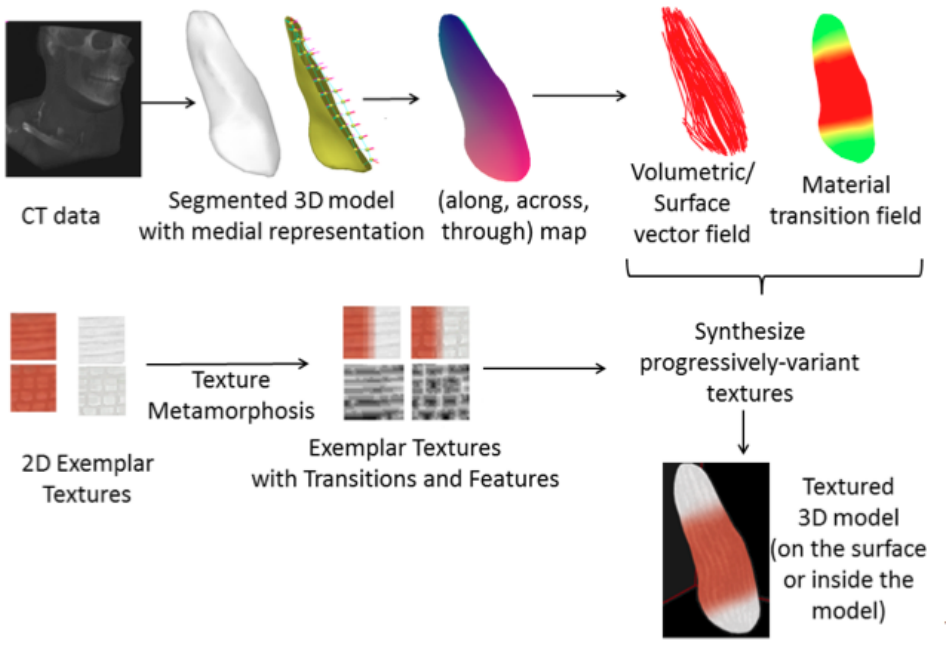

Figure 1.9: Pipeline of the model-guided texture synthesis framework. The top row shows the steps for computing the model-based guidance information (vector field and material transition fields) from the medial representation. The bottom row shows the steps for obtaining the exemplar textures.

1.4 Thesis and Contributions

The texturing objectives described in the first section with the possible applications described in Section 1.1 can be accomplished by a set of methods with new underlying concepts, supporting the following thesis:

Thesis Statement: *Combining texture synthesis (in 2D and 3D) with volume parameterization provides a framework for generating illustration of patient-specific data. Also, structural and material variations of organs can be illustrated by textures obtained using a metamorphosis approach that takes intensity and structure into account to generate a smooth transition from one texture to another.*

This thesis is supported by descriptions of the methodologies and the results obtained using them. The methods generate the inputs of MGTS framework, texture metamorphosis and model-based guidance generation, and they create the patient-specific textures for the specific anatomical organs, surface and solid texture synthesis. The results obtained using these methods created patient-specific anatomical illustrations which enable the doctors to distinguish the anatomical structures from one another and to understand their shapes under radiation dose.

The methods developed, their underlying concepts, and the results form a set of contributions:

1. *A method of computing 3D model-based guidance information, such as an orientation field and a material transition field, for different instances of the same anatomical organs using a volumetric model-based coordinate system obtained from a medial representation.*

2. *A method of generating progressively changing 2D exemplar textures using a new energy-based metamorphosis approach. This method utilizes appearance information along with the structural features in a single framework to create transformations from a source texture to a target texture.*

3. *A method for constraining the synthesis of structural primitives in a solid texture using medial-based representations of the structural primitives (textons). The solid textures are synthesized from three 2D exemplar textures. The medial representations are generated from the 2D exemplar textures, and they are used in forming the synthesized solid texture.*

4. *A method for controlling the number of textons in a texture synthesized from exemplars using a texton-based histogram matching technique in texture synthesis.*

5. *A method of synthesizing progressively-varying, anisotropic, model-based surface and solid textures for obtaining "Netterly" renderings of patient-specific anatomical organs using a novel framework. This framework uses model-based guidance information and progressively-changing 2D exemplar textures as input and creates textured anatomical models as output. Depending on the application, the contributions in item 3 and 4 are used in texture synthesis.*

6. *Demonstrating that MGTS supports the illustration of any patient's segmented data so as to achieve illustrative visualization of his/her organs. This is useful to identify and convey the shape of the organs in a complex environment with many structures.*

7. *Demonstrating the usage of texture in radiation dose visualization to identify and convey the shape of organs whose surfaces are washed with dose colors.*

Within each section in this dissertation, I describe the validation for each method.

1.5 Overview of Dissertation

The rest of the dissertation is organized as follows:

Chapter 2 gives background material in illustrative visualization, image morphing, 3D volumetric vector field computation, textures, texture synthesis on surfaces, solid texture synthesis, and texture synthesis for medical illustrations.

Chapter 3 describes the novel approach for texture metamorphosis which allows one texture to morph into another.

Chapter 4 explains the model-based coordinate system used in MGTS and how it is used to compute the guidance information for texture synthesis.

Chapter 5 presents the techniques used in synthesizing anatomical textures on the surface(s) of anatomical models.

Chapter 6 describes my model-based solid texture synthesis approach in detail.

Chapter 7 presents conclusions, discussions, possible applications of MGTS, and suggestions for future work.

Chapter 2

Background

This chapter presents the background material from computer graphics, computer vision, and image analysis that is related to this dissertation. Section 2.1 gives an overview of the basic concept of texture. That includes texture definition, texture features, applications of textures in different fields, and texture mapping techniques with its applications in computer graphics. Section 2.2 presents the existing work on texture metamorphosis, which includes morphing one texture into another. Section 2.3.1 explains the basics of texture synthesis, which lays the background for the remaining parts of section 2.3. Existing work in this field is mostly focused on exemplar-based approaches. Specifically, section 2.3.3 gives an overview of texture synthesis on a surface in 3D, and it describes the existing methods. In addition, section 2.3.4 presents a survey of solid texture synthesis work. Section 2.4 presents an overview of surface and volumetric representation techniques, and it describes in detail the medial object representation, which is a deformable model-based representation used as a guidance for texture synthesis in this dissertation.

2.1 Texture

Texture describes the variation of the intensity (or hue, saturation, surface normal, opacity) of a surface or a volume, quantifying properties such as smoothness, coarseness

and regularity of the intensity pattern. Textures are usually described by qualitative structural features, such as fineness vs. coarseness, smoothness, granularity, directionality, roughness, regularity vs. randomness. Texture features are used as measurements to characterize a texture. They are important for many applications such as texture analysis, texture retrieval, texture classification, and texture synthesis (Materka et al. (1998), Manian & Vásquez (2003)). In this dissertation the descriptors that are used to extract important features for texture synthesis are presented in detail.

2.1.1 Texture categories

In most of the texture synthesis applications textures are categorized as structural vs. stochastic textures, as well as textures with both regular and stochastic aspects, such as natural textures:

1. Stochastic textures: They look like noise, such as sand and grass texture. Colored dots are randomly distributed in the image, and they don't have specific structure or organization.

2. Structural textures: They have regular patterns, such as the texture of a stone wall or floor tiles.

The textures in these categories and the transition between them is visualized in Figure 2.1.

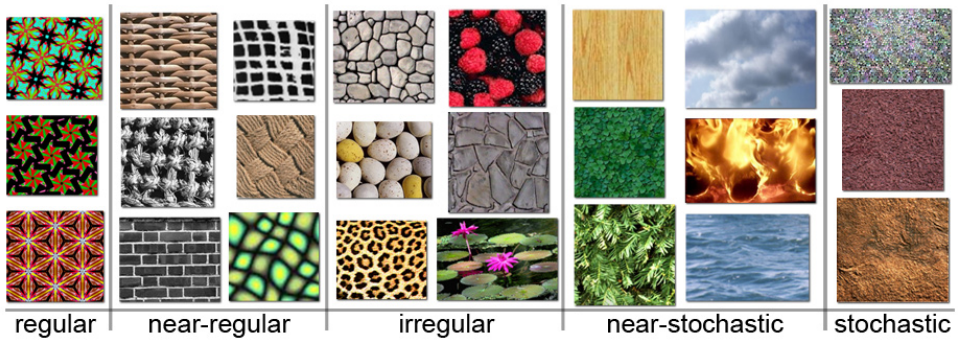

Figure 2.1: Texture spectrum presented in Liu et al. (2004).

Medical textures have variations in their features depending on the tissues in the anatomical organs. Realistic medical textures are mostly stochastic and near-stochastic (Figure 2.2). On the other hand, nonrealistic textures used in the medical illustrations have more regular patterns to express the important information in the tissues (Figure 2.3).

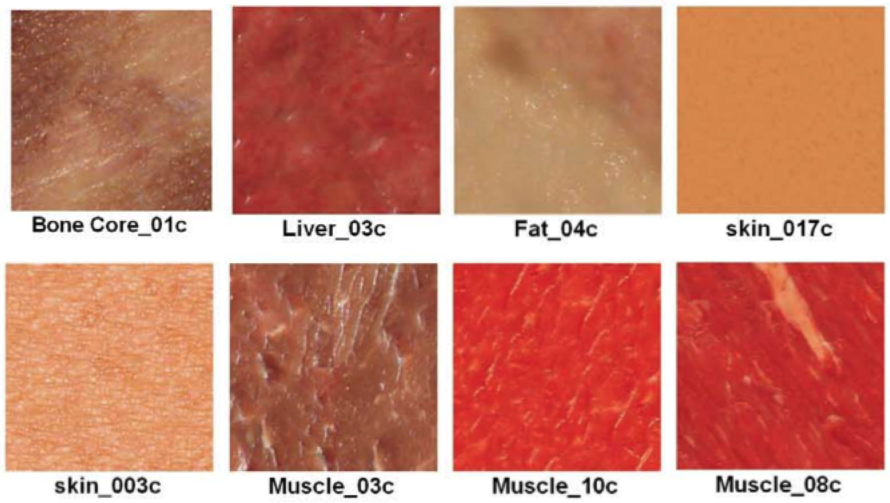

Figure 2.2: Texture samples from Dosch medical visualization database.

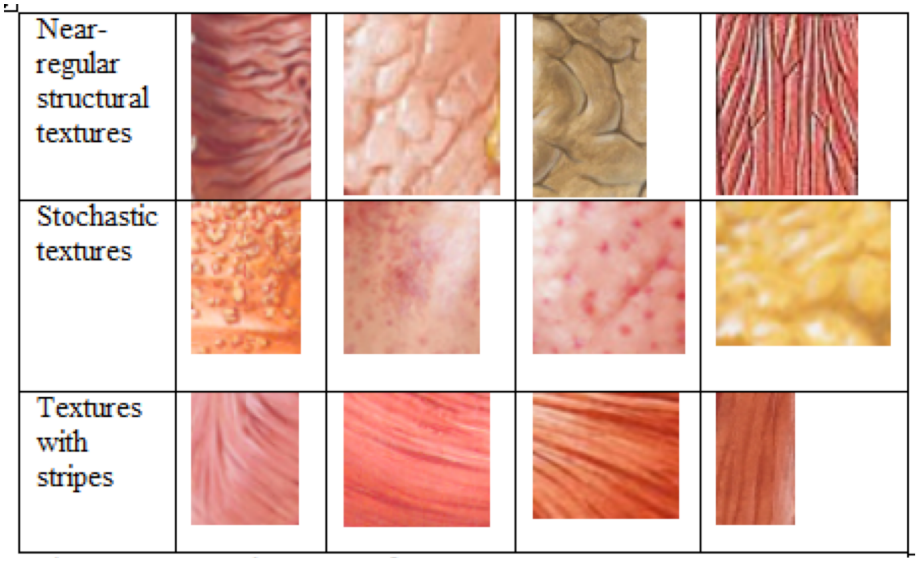

Figure 2.3: Texture samples from medical illustrations.

Textures can also be categorized as stationary or non-stationary depending on the variation of the texture parameters over space (Rosenberger et al. (2009)):

1. Stationary textures: They include a single texture.

2. Non-stationary textures: The texture or its parameters vary across the image.

In texture synthesis applications textures are also categorized based on their spatial dimensionality:

1. 2D color textures: Such textures are represented as 2D images. They have smooth, locally planar surfaces. Each pixel in the image can contain any number of intensity values.

2. 3D surface adjusting textures: Such textures are also represented as 2D images. One possible use of such textures is to create surfaces with varying roughness, in which the texture is related to local variations in the surface normal (local surface curvature). Normal maps, reflectance maps, and displacement maps are possible applications.

3. Solid textures (volumetric textures): Such textures are described in 3D space. They can be considered as a stack of images. They are used as space-filling textures for 3D models. It is possible to visualize both the exterior and the interior of models using them.

2.1.2 Texture features

Textures usually have complex changes in orientation, scale, or other visual appearance such as brightness and contrast. It is hard to describe all of the distinctive characteristics of a texture using a single mathematical representation. Due to the variety of textures, texture features are designed for each specific application area.

In this dissertation the texture features are categorized into four families as in Mirmehdi et al. (2008). This categorization is not crisp, and combinations of different categories can generate new features for different applications:

1. Statistical features: They measure the spatial distribution of pixel values by computing the statistical distribution of image intensities at specified relative pixel positions. They are characterized by their order, such as first-order statistics and second-order statistics. These features are used to analyze characteristics of texture, such as smoothness and coarseness.

   (a) Image histogram: This feature is a collection of first order statistical information, describing the frequency of appearance of particular color tuples in a region of the texture. It is easy to compute. It is invariant under rotation and translation, and insensitive to the exact spatial distribution of color pixels. In addition to the histogram of intensity values, a histogram of interesting features (e.g., features extracted by a bank of linear filters (texton histograms)) can be used for describing the texture (Dana & Nayar (1998)).

   (b) Gray level co-occurrence matrix: This feature represents second-order statistics of gray levels in pairs of pixels in an image (Haralick (1979)). It stores the frequency of every particular pair of gray levels in the pixel pairs. Various statistical properties which serve as textural features, such as energy and entropy, can be obtained from the co-occurrence matrix.

   (c) Local binary patterns: This feature can be considered as a higher order statistical feature. It represents the texture by a binary image which is created by labeling the pixels in that texture (Ojala et al. (2002)).

2. Structural features: They characterize texture as being composed of texture primitives which are regularly arranged on a surface according to some rules or guidelines. These features can be used in both texture synthesis and texture analysis. Pixel intensity properties in the primitive and the spatial relationship between primitives are considered for describing the structural features of a texture. The computation of structural features requires two main steps:

(a) Extraction of texture primitives: Texture consists of texture primitives (texture elements) called textons (Julesz (1981)). A texture primitive can be an individual pixel, a region with uniform values, or line segments. It usually has some tonal and/or regional shape properties. The number and size of the primitives in a texture is also important for describing the textures.

(b) Modeling and generalization of the spatial placement rule: This can be done by learning the statistical properties of the distribution of the primitives in space or finding the geometrical relationship between them (Lu & Fu (1979)).

3. Signal-processing-based features: Such features are extracted by applying filter banks to the image and computing the energy of the filter responses (Grigorescu et al. (2002), Bovik et al. (1990)). Based on the domain (spatial or frequency domain) they are applied in, they can be categorized into two types:

   (a) Spatial domain operators, such as Sobel, Robert, or Laplacian, are used to filter the images to extract features like edges, lines, or isolated dots.

   (b) Frequency domain filters like Ring and Wedge are used to extract texture features when the associated kernel is difficult to obtain in the spatial domain.

4. Model-based features: Stochastic, generative models are used to represent images in which the model parameters capture the essential texture features.

   (a) Fractal models: Such models are mostly used for representing natural textures which have a statistical quality of roughness and self-similarity (Chaudhuri & Sarkar (1995)).

   (b) Random field models: Such models consider local information in order to achieve a good global image representation. Markov Random Fields (MRF) are one example (Cross & Jain (1983)). The Markov random field is a conditional probability model allowing one to model local spatial interactions

among entities, such as pixels. It is used in many applications, including texture synthesis and texture classification. This model assumes that the intensity at each pixel in the image depends on the intensities of only the neighboring pixels. It measures the distribution of intensities in pixel neighborhoods.

2.1.3 Sources of texture

Textures can be obtained through various imaging devices such as cameras, microscopes, and computed tomography (CT) and magnetic resonance (MR) imagers. There exist many texture databases that researchers use to test their methods, such as the Brodatz collection (http://www.ux.uis.no/ tranden/brodatz.html), and the Curet database (http://www.cs.columbia.edu/CAVE/software/curet). The Dosch medical visualization database (Design (2001)) contains textures for the depiction of organs, ligaments, bones — everything that needs to be visualized for medical applications (Figure 2.2). These images can be used for medical animations, educational material, presentations of research results, and the creation of informative illustrations. The database contains 140 high resolution, seamlessly tileable textures.

In addition, hand drawn illustrations and drawings mostly composed of textures can be used for constructing a texture database. In this dissertation input textures are mostly obtained from medical illustrations in (Netter (2009)) though other sources can be used depending on the purpose of the application:

- Manual art: Different painting techniques can be used for designing textures. The advantage of this method is that it is flexible and easy to control. On the other hand, it needs expertise in painting techniques.

- Procedural methods: Mathematical functions can be used for creating textures. This method is efficient and flexible. However, it is not general, and it requires ex-

pertise in mathematics. Finding a function and tuning its parameters for creating a specific texture is a challenging task.

- Photographs: Taking pictures of real world environments and cropping the regions in them using imaging tools is another way of obtaining textures. This method makes it possible to obtain many realistic textures. However, controlling the features in the acquired textures is not possible in this method.

- Texture synthesis: Bigger texture patches can be synthesized from smaller texture samples using texture synthesis methods. The details of texture synthesis are explained in Section 2.3.

2.1.4 Applications of textures

Textures have been used in many applications for different purposes:

1. **Texture in visualization**: Texture is important for visualization in many ways. It helps us to understand the orientation, shape, and spatial layout of a surface. In addition, it can be used to code information. In many visualization applications different channels of textures are used to visualize different features of data. For instance, Gabor functions can be used to produce distinct textures for the underlying data using orientation, scale, and contrast dimensions of texture for encoding information (Ware (2004)). Moreover, textures can be used to visualize the material characteristics of different objects. For instance, textures can be used to visualize the underlying features in the seismic data (Patel et al. (2009)). Besides these, texture has various applications in visualization, such as flow field visualization, multidimensional data visualization, and illustrative visualization.

2. **Texture in computer graphics**: In computer graphics, a texture can be a bitmap image or a procedural function. It refers to the digital representation of

the surface of an object. In addition to 2D qualities, such as color and brightness, textures can store 3D properties, such as in displacement mapping. In a virtual environment once a texture has been defined, it can be used for rendering a 3D model by mapping it onto a surface geometry to enhance realism in the scene. This process is called texture mapping. Different texture mapping techniques exist for applying 2D or 3D textures to a 3D model.

2.1.5 Texture mapping

Texture mapping is the mapping of an image onto a surface or within a volume in 3D. In computer graphics, texture mapping is used to enhance visual richness in scenes, such as making objects look like natural materials (e.g., marble, wood) by applying a pattern of color onto the object, or adding 3D detail to the surface by altering the surface details of an object so that it appears rough.

Texture mapping has two important components. One deals with how to warp a texture onto a surface or within a volume, and the other deals with how the discrete pixelization of the texture should be handled in order to avoid aliasing. In this dissertation I will cover the first component in detail.

In computer graphics, textures can be represented as rectangular 2D or 3D arrays of data. Each entity in this array can store color data, luminance data, or color and alpha (opacity) data. The difficulty in texture mapping comes from having to map a rectangular texture to a nonrectangular object (region), because the texture might get distorted. For each point on an object, which point in the texture it corresponds to must be found (Figure 2.4).

Understanding coordinate systems is important for understanding how the mapping is achieved (Wolfe (1997)):

1. Texture coordinates: These coordinates are used to specify the points in the image to be mapped to points on the object. Texture space is labeled with (u, v)

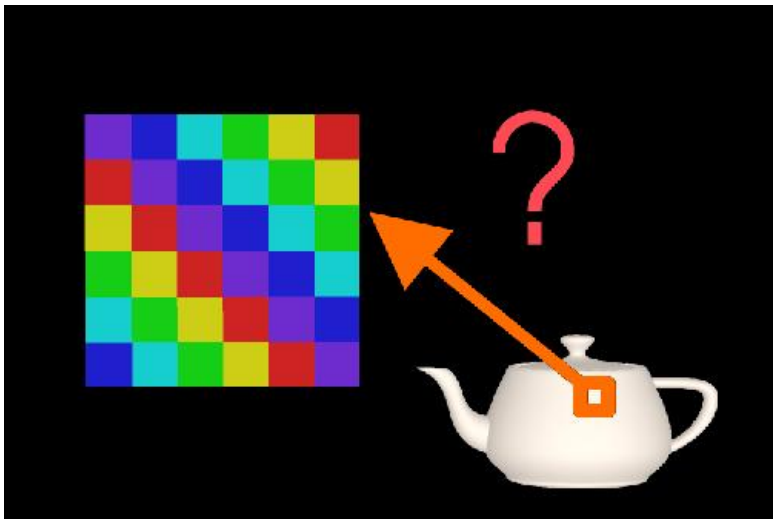

Figure 2.4: 2D texture mapping. (Figure is taken from Wolfe (1997).)

coordinates. Every vertex in a 3D model must be given a texture coordinate. This is used to find the texture location associated with that vertex.

2. Object and world coordinates: A world coordinate system covers the simulated world (the scene to be rendered). It is the base reference system for all the models inside a scene. On the other hand, object coordinates cover the object itself. When an object is created in a scene, a point should be picked as the origin of that particular object, and the orientation of the object to a set of model axes should be determined as the model-based axes. World and object coordinates are related with where and how the mapping takes place. This mapping is categorized based on whether the object is fixed with respect to the texture, or the texture is fixed with respect to the object (Figure 2.5).

   (a) World coordinates: The origin and coordinate axes of the texture remain fixed in the world coordinates. This coordinate system is not used so widely since the texture is mapped onto a surface in world space. This causes pattern shifts on the objects when the object moves through the space.

   (b) Object coordinates: The origin and coordinate axes remain fixed relative to an object independent of object's position and orientation. Most mapping

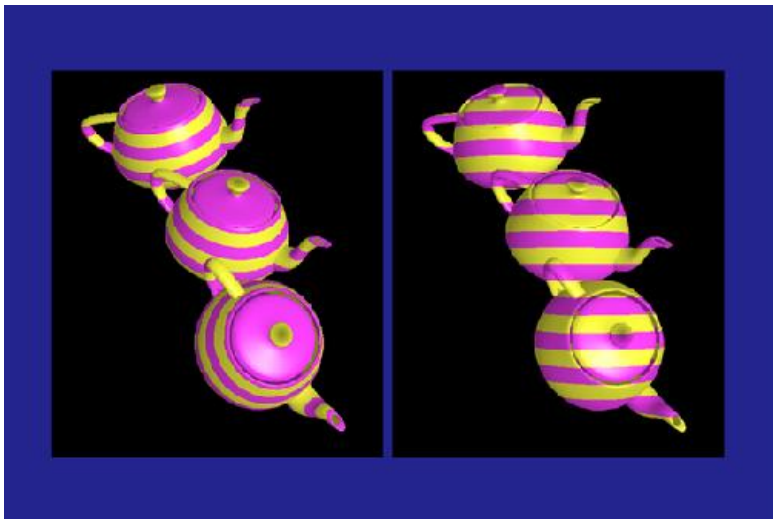

Figure 2.5: Texture mapping is done in (left) object coordinates, (right) world coordinates for teapot models in different positions. (Figure is taken from Wolfe (1997).)

techniques use this coordinate system in which the source image (texture) is mapped onto a surface in 3D object space. This means if the handle of a cup is painted red, it will remain red even when the orientation and position of cup change.

There are various ways to map a 2D or 3D texture onto a surface. These approaches can be divided into 2D and 3D techniques, which both deal with how the color of each pixel (in 2D) or voxel in (3D) is going to be found in the texture:

1. 2D texture mapping techniques: These techniques refer to placing a 2D image onto an object using methods similar to pasting wallpaper onto a bumpy wall. The texture is defined in 2D, and it is mapped onto a 2D or 3D object. One would like to avoid distortion of the mapped textures, and seams between textured regions when using these techniques. There are many approaches proposed for 2D mapping:

   (a) Project the texture: This method, presented in (Bourke (1996)), consists of a map shape which is used to convert the (x, y, z) value from the object to the coordinates of map shape. The map shape can be planar, cylindrical, spheri-

cal, or box. For instance, spherical mapping converts the (x, y, z) coordinates to spherical coordinates (θ, ϕ). After these coordinates are obtained, they are transformed into rectangular texture coordinates, which are mapped over the surface of the sphere (Figure 2.6).

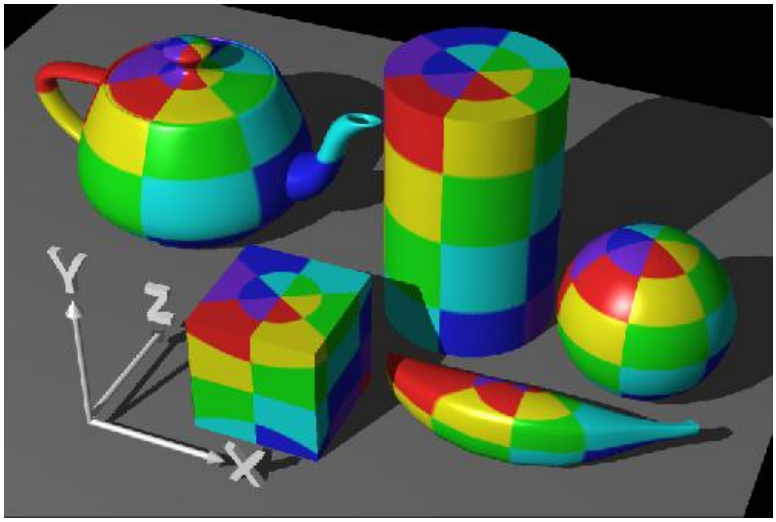

Figure 2.6: The objects have a map shape of a sphere, and the poles of the sphere are parallel to the y-axis (Wolfe (1997).)

(b) Unfold the surface: This method consists of unfolding the 3D mesh at the seams and automatically laying out the triangles on a flat page. This process can be done by 3D modelers or using automatic techniques.

(c) Tile the textures: This method consists of representing a large texture as a small set of tiles and combining these tiles into a large texture. In this method it is important to know which tile is used for each region of the texture (Wei (2004)).

(d) Make an atlas: A texture atlas is an efficient color representation, which consists of many smaller sub-images (Levy et al. (2002)). Each sub-image is a texture for some part of the 3D model.

2. 3D texture mapping techniques: These techniques refer to placing a 3D image onto an object using methods similar to either carving marble to create an object

or warping the marble to fit into an object. The texture is defined in 3D, and it is mapped onto a 3D object. For each point on the object, we must find the corresponding point in the 3D texture. There are two approaches for 3D texture mapping:

(a) World-based mapping: This technique is analogous to carving the object from a block of 3D texture. Texture is considered as a block which objects fit into. Each renderable point (x, y, z) in world coordinates inside the 3D model determines its color without the use of any intermediate transformation. This technique is straightforward and works mostly for isotropic textures. It is possible to avoid distortions and seams in this technique.

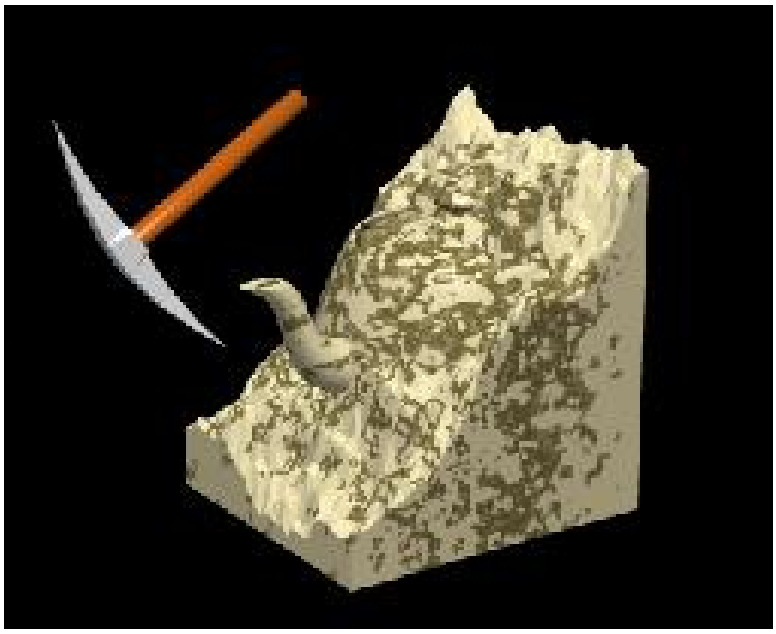

Figure 2.7: World-based texture mapping for 3D textures. (Figure is taken from Wolfe (1997).)

(b) Model-based mapping: This technique is analogous to warping a block of 3D texture to fit into an object. 3D textures can be clipped and transformed based on some guidance information. This can be a user-defined volumetric tensor field as in (Takayama et al. (2007)) or a model-based coordinate system as proposed in (Merck (2009)). In Takayama et al. (2008b), texture mapping

is achieved by cropping and warping cubic solid textures based on the user input. In that method, first the triangular mesh is converted to a tetrahedral mesh model and a 3D texture coordinate is assigned to the vertices of each triangle in each tetreahedra. Then these coordinates are used for mapping the textures inside the model (Figure 2.8).

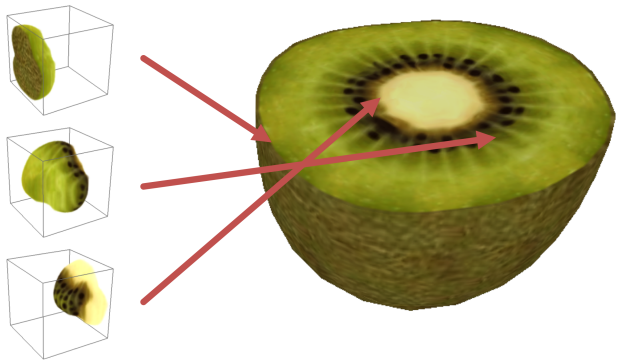

Figure 2.8: Model-based texture mapping for 3D textures. (Figure is taken from Takayama et al. (2008b).)

There are many difficulties in texture mapping:

1. Aliasing: This problem is usually apparent with regular textures. It can occur any time we resample a texture, for instance when we scale it, rotate it, or map it onto a 3D model. Mipmapping, bilinear filtering, trilinear filtering, and anisotropic filtering are some of the solutions to aliasing.

2. Singularities: Objects of spherical topology have at least one singular point on the surface mesh. For instance, the meridians on the earth have two singularities: one at the North Pole and one at the South Pole. It is impossible to define the second texture coordinate at these points. These points should be anticipated and handled as special cases in texture mapping.

3. Distortions: Since mapping a 2D image onto an arbitrary 3D model requires transformation of the texture, there will be regions in which the texture looks distorted.

4. Seams between texture patches: When multiple textures are mapped on the surface of a mesh, the transition between the tile boundary regions should be smooth.

2.1.6 Texture layering

Layering and blending of multiple textures are often applied in computer graphics to create realistic looking models (e.g., in terrain rendering). Multiple textures can be combined onto the same surface through a variety of combination rules. For instance, we can combine different textures on a mesh to visualize the transition between regions. The triangles in these regions would have 2 sets of texture coordinates, and they have to be interpolated. This interpolation can be achieved using multi-pass rendering or programmable GPU architectures.

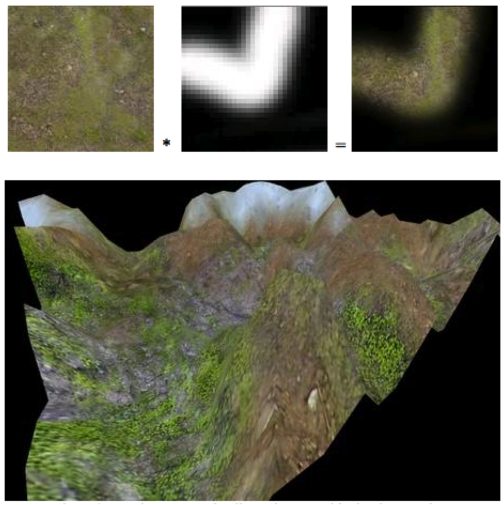

Figure 2.9: (Top row) Texture blending using alpha maps; (bottom row) terrain texture blending (`http://www.jenkz.org/articles/terraintexture.htm`).

Texture layering is widely used for terrain rendering to create the illusion of a rich landscape. This is achieved by overlaying different textures on top of each other by considering the height of the terrain mesh at each point. Alpha blending is widely used for layering these textures (Figure 2.9). While in terrain rendering alpha maps are computed based on the heights of the terrain mesh, they can be computed based on

other constraints depending on the application.

2.2 Texture Metamorphosis

Texture metamorphosis refers to the process of smoothly interpolating between two textures, T_0 and T_1 (Figure 2.10). Transformation of the intensity values along with the structural features (e.g., patterns, textons) in the texture is an important contraint that should be taken into account in texture metamorphosis. This is important in order to achieve smooth transformation of patterns along with the intensity values and makes texture metamorphosis problem different from image metamorphosis problem. Several approaches have been proposed in the literature to find the transformation from one texture to another. Most of these techniques are based on blending structural features of two input textures (e.g., feature images such as edges, thresholded binary images, signed distance fields) and synthesizing output textures using this blended feature as a constraint. The technique proposed in this dissertation differs from these techniques by considering the change in structural features and intensities in a single framework without any need for texture synthesis.

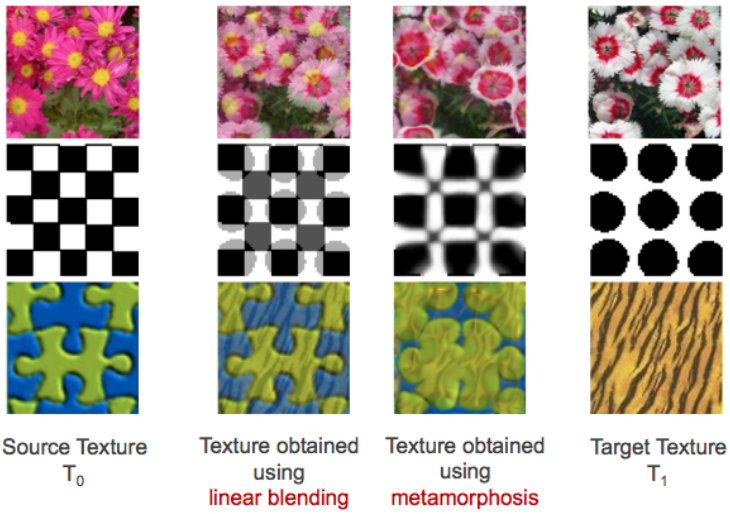

Figure 2.10: (Left) Source texture T_0 (input texture); (middle two) interpolated textures (output texture); (right) target texture T_1 (input texture).

The first approach for morphing structural textures was presented in (Liu et al.

(2002)) using a pattern-based approach in which the user specifies landmarks and their correspondences for two textures. The landmarks are used to create a warp field, and cross-dissolving accommodates texture appearance changes. This approach assumes that texture images are composed of similar texture patterns. In addition, it requires a large amount of user interaction. Later, in (Wei (2003)) an algorithm for creating morphed textures between two input textures using texture synthesis methods is proposed. In this method, mixed textures are synthesized using the given input textures by constraining the texture synthesis via user-defined weighting images.

Zhang et al. (2003) synthesized progressively variant textures using a texton mask as the feature image for input textures. In their method first a mixed feature mask is generated by blending and binary-thresholding the textons in the feature images. Then this mixed feature mask is used as a constraint in texture synthesis in order to generate the mixed output texture. The main disadvantage of this approach is that it does not guarantee a smooth transformation from one image to another. In addition, an arbitrary number of images between two input images cannot be generated.

Liu et al. (2004) proposed a system to design novel textures by analyzing and manipulating textures. Their method allows the user to warp texture features which can be used for alignment in morphing. User-defined feature maps are also used in (Matusik et al. (2005)) to produce warps between textures. They also proposed a histogram matching technique to avoid blurring of features. However, the approach of Matusik et al. (2005) is only applicable to textures that are similar to each other.

A level set advection method for morphing between texture features was proposed in (Ray et al. (2009)). This method first finds a warp that takes texture features from one image to the corresponding features in the other texture. Then a constrained texture synthesis approach is applied to generate a texture with a given feature image. Recently a new patch-based approach was proposed in (Ruiters et al. (2010)). In this approach individual neighborhoods of the input textures are locally warped and blended, and then

they are used to create new textures by an optimization algorithm. The disadvantage of this patch-based approach is that it cannot handle materials with large differences in feature scales. In addition, since the warping is done using only feature information, it cannot handle textures that do not have clear, meaningful features.

The metamorphosis formulation that the approach proposed in this dissertation builds on has been studied for example in (Trouve & Younes (2005); Holm (2009)). It also builds on numerical solution methods that have been proposed in (Miller & Younes (2001) and (Garcin & Younes (2005)). That line of work has mostly been concerned with morphing gray-valued or color images and has so far not dealt with the problem of texture morphing.

2.3 Texture Synthesis

Texture synthesis is the process of generating a large texture either from a small exemplar texture using its stochastic and structural features or using functions to generate patterns and colors. There are two main challenges in texture synthesis; both are related with scale. The first is to capture the input exemplar texture's visual characteristics while adding enough randomness to maintain its natural feel. The second is to maintain the relationship between the texture features, such as relative position, from the exemplar texture. Synthesizing textures based on target-specific constraints, due to properties such as orientation, scale, and region, also brings new challenges to the problem.

Texture synthesis is used in many fields in computer graphics and digital image editing tools. It has been widely used in surface and scene rendering to make objects look more realistic. Other applications include image completion and video synthesis.

There have been two main approaches proposed for 2D and 3D texture synthesis: procedural and exemplar-based.

1. In procedural approaches the textures are created using functions that represent

"randomness" in nature. Depending on the functions and the parameters used in them, realistic representations of natural elements can be obtained (e.g., wood, marble, granite, metal, stone). Perlin noise is most commonly used as basis functions in this method (Perlin (2002)). However, other simple functions are also used, such as the sum of sinuisoidal functions. More details about procedural texture synthesis can be found in Ebert et al. (2002).

The procedural approach has many advantages:

(a) It allows the synthesis of 2D and solid textures at any resolution.

(b) The memory needed to save the procedure is very small, compared to saving a whole texture. For instance, saving a solid texture in memory is very inefficient since we have to store the value of each voxel in memory. In contrast, the color values for each voxel (x, y, z) on the surface of the mesh can be easily computed using a function during rendering without any need for precomputation and storage.

(c) It generates 2D and solid textures that are unaffected by distortions of surface parameterization since each point on the surface is generated based on a locally-flat 2D mesh coordinate.

(d) It handles the continuity between the adjacent surface patches automatically using functions, which makes textures remain consistent on the model.

The procedural approach also has two important problems:

(a) It is not general. It is hard to find the appropriate function and the parameter values for it to match a particular exemplar texture.

(b) It generates natural textures which have a cleaned appearance. This makes it look artificial. Artificial noise can be added to solve this problem.

2. In exemplar-based approaches the textures are created using one or several input textures (2D or 3D depending on the approach). The aim is to generate an output texture that matches the given input textures. The most successful approaches in this technique are based on Markov Random Fields (MRF), though techniques using different texture models have also been proposed. In MRF-based methods the aim is to generate a texture in which the spatial neighborhood of each pixel is similar to at least one neighborhood in the input texture (Paget & Longstaff (1998)).

   Exemplar-based techniques have many advantages:

   (a) They can replicate both stochastic and structural features of a given texture.

   (b) No expertise is required for setting their parameter values.

   (c) They can be combined with guidance information, which can come from the underlying data or surface model, to generate a constrained texture on the surface of a model.

   On the other hand, these methods have many disadvantages:

   (a) They only work well for input textures that are homogeneous.

   (b) They need large memory for storing the output texture (compared to procedural methods) since they initially synthesize output texture and then map it onto the model.

   (c) They are not time-efficient when compared to the procedural approach. This is caused mainly by the large number of neighborhood comparisons in the search phase — this phase will be explained in the next section.

In this dissertation exemplar-based techniques are used for synthesizing anatomical textures both on and inside the surface. Further background on this topic is presented in detail in next sections.

2.3.1 2D texture synthesis

There are pixel-based and patch-based techniques proposed for synthesizing textures. Pixel-based methods go in order through all pixels of the synthesized texture and use the input pixel with the best matching local neighborhood (Efros & Leung (1999)). Patch-based methods copy the best-fitting parts from the input texture into the synthesized texture and append these parts to each other (Efros & Freeman (2001), Kwatra et al. (2003)). There are also optimization-based (Kwatra et al. (2005)) and hybrid approaches (Nealen & Alexa (2003)) that combine the advantages of pixel-based and patch-based methods. In this section the algorithms in these categories are described briefly. Details of the methods can be found in the cited papers in each section.

2.3.1.1 Pixel-based texture synthesis

In pixel-by-pixel synthesis first each pixel in the output texture is initialized to intensity of a random pixel selected from the input texture. Then the output texture is synthesized by considering each pixel in a specified order (Figure 2.11).

The techniques in this category are composed of a search phase, in which an appropriate section of the exemplar is found as a best match, and a copy phase, in which the selected pixel from the exemplar is copied to the output texture:

1. Search phase: For each pixel in the output texture, a spatial neighborhood around it is considered. Then for this neighborhood the best match or the best N matches are found from the neighborhoods in the input image. The sum of squared differences of pixel intensities are used as a matching function to find the best matches. There are many factors that are considered in different approaches during this phase:

   (a) Order of synthesis: In most texture synthesis algorithms raster-scan ordering is used. On the other hand, for constrained texture synthesis, such as syn-

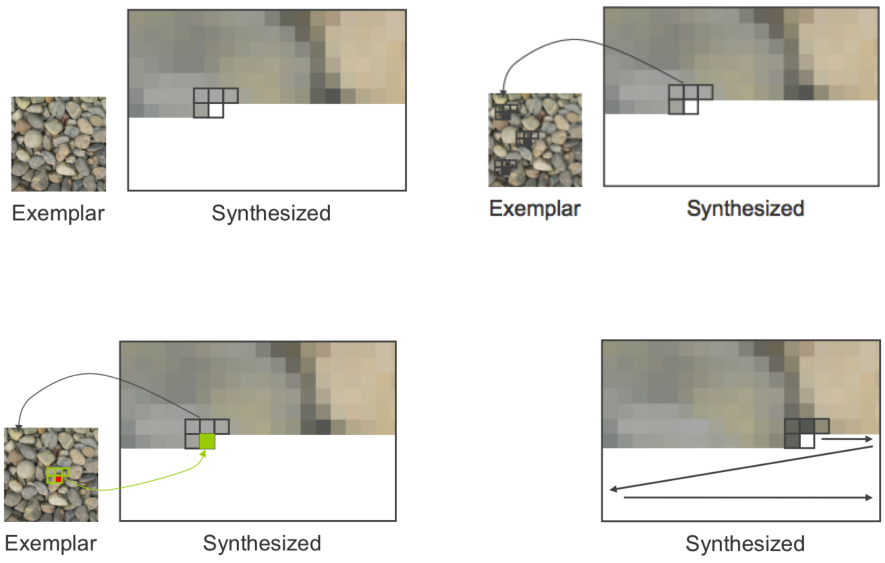

Figure 2.11: The search phase in exemplar-based texture synthesis. (Top left) An L-shaped neighborhood is created around the pixel that is going to be synthesized in the output (synthesized) texture. (Top right) Best matches are found in the exemplar texture (input texture). (Bottom left) The color value of the selected pixel is assigned to the synthesized texture. (Bottom right) The synthesis is done in scanline order.

thesis of oriented or region-specific textures, this order can be specified based on the orientation field or material transition field. In certain situations with irregular meshing, random ordering can be used.

(b) Size and shape of neighborhoods: Since the Markov random field model considers the neighborhood information in texture synthesis and we would like to preserve structures in the output texture, the size and shape of the neighborhood are important. The size should be as large as the largest texture structure in the input image, and the shape should include the already synthesized pixels in the output texture. The size is usually specified by the user since it is hard to compute the size of the structures in the input texture automatically. In addition, an L-shaped or a rectangle-shaped neighborhood is usually used. If raster-scan ordering is used when synthesizing output texture,

an L-shaped neighborhood is used in the first pass and a rectangle-shaped neighborhood is used in the next steps.

(c) Multi-resolution neighborhoods: These are used to reproduce large scale texture elements without having to use a very large neighborhood by operating at multiple resolutions. For instance, for synthesizing a $2N \times 2N$ output texture, we first reduce the size of the input and output textures to $N \times N$ pixels and synthesize output. This captures the large scale elements. Then, we expand them both to $2N \times 2N$ and use the $N \times N$ synthesis output as a starting point. Using this approach we can preserve the structures in an efficient way. Image pyramids are usually used for achieving multi-resolution.

2. Copy phase: In this phase the color of the pixel is assigned using the results obtained in the search phase. If only one best match value is found, this value is copied to the output texture. On the other hand, if N best matches are found, a pixel from this set is selected randomly and its value is copied to the output image.

The main advantages of the pixel-based approach are that it is simple and gives reasonable results. On the other hand, there are failure cases, and it is hard to control. In addition, it is slow. For solving quality and speed problems in texture synthesis, coherence-based techniques have been found very useful. The first method in this area was proposed in Ashikhmin (2001) and later was extended by Tong et al. (2002) for k-coherence. In k-coherence method for each pixel in the output texture k pixels with similar neighborhoods are found from the input exemplar texture. In this approach the search and copy phases mentioned above are the same except the following two additions:

1. Preprocess step: This step consists of construction of a precomputed set for each pixel containing the other pixels with similar neighborhoods.

2. Restriction on search: The search phase applies an additional restriction, which is the consideration of the already synthesized neighborhoods' precomputed sets.

K-coherence has been found very useful for improving the quality of the output textures, especially for natural textures.

2.3.1.2 Patch-based texture synthesis

Although patch-based and pixel-based approaches are similar to each other in the search and copy phases, there are many differences between them:

1. In patch-based synthesis patches are searched and copied instead of pixels.

2. In patch-based synthesis there is an additional step after the copy phase for handling the overlapped regions between the copied patches. Different approaches are proposed for this step:

   (a) Overwriting new patches over existing regions, as proposed in Praun et al. (2000), which works well for stochastic textures.

   (b) Blending the overlapped regions of new patches and existing regions (Liang et al. (2001)), which may cause blurring.

   (c) Finding an optimal cut through the overlapped regions, using dynamic programming techniques (Efros & Freeman (2001)) or graph cut approaches (Kwatra et al. (2003)).

   (d) Warping the patches to ensure pattern continuity across the patches (Soler et al. (2002) and Wu & Yu (2004)).

Patch-based approaches are fast and give satisfactory results for many textures. However they lack flexibility, and it is hard to speed up the process using parallelism. As in pixel-based approaches, tuning the parameters, such as neighborhood size and pyramid resolution, is required for high-quality synthesis.

2.3.1.3 Optimization-based texture synthesis

The advantages of pixel- and patch-based techniques are combined in a single-optimization-based algorithm proposed in Kwatra et al. (2005). This algorithm searches for the best match for each pixel using its spatial neighborhood as is done in the pixel-based approach. Then it uses a global optimization technique which resembles the patch-based approach to compute the value of each pixel in the copy phase.

The goal of this approach is to minimize a global texture energy function E that measures how much the synthesized texture deviates from the exemplar texture using the local 2D neighborhoods. By setting $\mathbf{e}$ as the input exemplar, $\mathbf{x}$ as the synthesized texture, $\mathbf{x}_p$ as the neighborhood of p, and $\mathbf{e}_p$ as the exemplar neighborhood closest to $\mathbf{x}_p$, the global energy function is described as follows:

$$E_t(\mathbf{x}; \{\mathbf{e}_p\}) = \sum_{p \in \mathbf{x}^t} ||\mathbf{x}_p - \mathbf{e}_p||^r. \tag{2.1}$$

Kwatra et al. (2005) found that the exponent r=0.8 makes the optimization robust.

The approach begins with a 2D image where the value of each pixel is randomly chosen from the exemplar, and it synthesizes the output texture by solving the energy function using an Expectation Maximization (E-M)-like algorithm. During synthesis, the two steps of the E-M-like approach are repeated until the energy is minimized:

1. E (expectation) step: This step can be considered as the copy phase of texture synthesis. In this step the value of each output pixel $\mathbf{x}_p$ is assigned using a re-weighted least squares approach, which considers the neighborhoods of $\mathbf{x}_p$ while it is computing the value of that pixel. More details of how it is computed can be found in (Kwatra et al. (2005); Kopf et al. (2007)). In that step the set of matching input neighborhoods $\mathbf{e}_p$ remains fixed. This step is also considered as the optimization phase.

2. M (maximization) step: This step can be considered as the search phase of texture synthesis. In this step the set of output pixels $\mathbf{x}_p$ remains fixed, and the set of best matching input neighborhoods $\mathbf{e}_p$ is found using a tree search.

Although this technique gives satisfactory results for many textures, it has some problems:

1. It is slower than the other techniques since it uses an iterative optimization.
2. The hierarchical tree search in the M-step is a speed bottleneck in the solver.
3. The least-squares solver in the E-step can cause blurring since it performs a weighted average of calculated input neighborhoods for assigning the value on the output.

2.3.2 Feature-guided (controllable) texture synthesis

The algorithms summarized in Section 2.3.1 are for synthesizing stationary output textures given a stationary texture as input. However, natural textures are mostly non-stationary. They usually have changes in their orientation, scale, or material characteristics depending on external or environment factors.

The textures that are local but non-stationary are referred as globally variant textures and can be synthesized using different methods:

1. User-constrained texture synthesis: The texture is synthesized using an input texture, and a control map given by the user.
   (a) Ashikhmin (2001) proposed an approach using inputs made of a local and stationary texture and a selection map for guidance. The selection map specifies choices in the input texture patterns for different regions of output texture; it is used as a constraint in texture synthesis.

(b) The approach proposed in Efros & Freeman (2001) is similar to Ashikhmin (2001) except that this method operates on patches instead of pixels.

2. Image-constrained texture synthesis: The texture is synthesized by considering the correspondence between the given two texture samples, A and A'. This correspondence is then used as a constraint in generating new texture B' from the other given texture B (Figure 2.12). In the papers Hertzmann et al. (2001) and Ramanarayanan & Bala (2007), the results of these approaches are shown with various textures. However, these techniques have two main disadvantages. The image similarity between two images is hard to specify, and the quality of the results depend on the synthesis and correspondence algorithms.

Figure 2.12: A new analogous image B' that relates to B is computed in the same way as A' relates to A. Here, A, A', and B are inputs to the algorithm, and B' is the output. Figure is courtesy of Hertzmann et al. (2001).

2.3.3 Texture synthesis on surfaces

One of the applications of texture synthesis is creating textures on the surface of objects to make them look realistic by illustrating their material characteristics. Given a surface mesh (or description) and an exemplar texture, the purpose is to synthesize texture on the surface without having distortions and discontinuities across the texture patterns. Many approaches (Turk (2001), Wei & Levoy (2001), Gorla et al. (2003), Praun et al. (2000)) for achieving this have been proposed in the last decade. The synthesis quality of these approaches depends on the underlying texture synthesis algorithm (e.g., non-

parametric sampling technique) and the surface handling scheme (e.g., subdivision of the surface into patches). The approaches usually consist of the following steps:

- Specifying the guidance information, such as orientation and scale of texture;
- Synthesizing texture on the surface;
- Rendering the surface with the synthesized texture.

2.3.3.1 Computation of guidance information on the surface

Although output textures can be isotropic, typically they can be synthesized according to the following guidance information specific to the model:

1. Scale: Texturing objects that have variations in the scale of the texture features at different locations on the surface (such as the size of giraffe spots differing on the animal's legs and body), can be achieved by computing a scale field on the surface mesh and considering it in texture synthesis. A scale field over the surface is usually constructed by getting its values at a few locations on the surface from the user and interpolating the values over the surface.

2. Orientation: Synthesizing anisotropic textures in which texture features smoothly orient along the underlying orientation field is important for texturing different objects, such as woven fabrics, bricks, animal strips, and wood grain. There are three general approaches for constructing the orientation field on the surface: utilizing constant or principal curvature directions at the vertices (Gorla et al. (2003)); using relaxation methods (Wei & Levoy (2001)), or using user input (Praun et al. (2000)).

3. Material: As in the scale field on the surface, material transition can be obtained from the user utilizing an interface for specifying the regions with different materials.

2.3.3.2 Synthesizing texture on the surface

There are many approaches proposed for synthesizing texture on the surface. They can be categorized based on how they handle the surface during texture synthesis:

- Dense points on the surface: This approach populates the surface with a dense set of points, in which each point is treated as a pixel. The creation of the point is usually done using point repulsion, which distributes points on the surface randomly and then finds their final position by treating them as particles with the same charge. Then a pixel-based texture synthesis approach is applied to synthesize color for each point on the surface. As in the other texture synthesis approaches, texture synthesis can be done at multi-resolution from coarse to fine. This speeds up the process and increases the quality of the textures.

  Synthesizing texture by coloring the vertices has been used in (Turk (2001) and Wei & Levoy (2001)). The method presented by Turk consists of two steps for initialization and one step for synthesis. In the first step a hierarchy of points from low to high intensity is created on the surface of the mesh, and these points are connected with each other to construct a hierarchy of meshes. In the second step the user draws the direction of the texture at several locations, and the vectors are interpolated over the remainder of the surface. In the third step the outputs from the first and second steps are used for synthesizing texture. The main problem in this approach is that it requires an additional mesh resampling step.

- Surface unfolding into the plane: This approach requires unfolding the mesh onto the plane, synthesizing the texture on each flattened patch in 2D, and mapping it back onto 3D surface. Unfolding the mesh includes subdividing the mesh into patches and mapping them onto 2D surfaces. Subdivision of the surface can be done using different techniques depending on the representation of the model. In Ying et al. (2001) the models are represented as subdivision surfaces, making the

texture chart creation straightforward. On the other hand, in Gorla et al. (2003) the models are represented polygonally, making it necessary to create large charts to achieve minimal distortion.

- Triangle patches: This approach is based on treating each triangle or a set of triangles on the surface mesh as patches Praun et al. (2000) and Li et al. (2008). First, for each triangle or a set of triangles, per-vertex texture coordinates ((u, v) coordinates) are computed, and a texture map is specified. Then texture is synthesized on this texture map in 2D. At runtime, triangles on the 3D mesh are rendered using the 2D texture maps.

2.3.4 Solid texture synthesis

While 2D textures are commonly used for visualizing the object's external surface, solid textures are used for providing volumetric information about the objects' interior appearance. Solid textures can be obtained by acquiring real data using 3D cameras (such as CT) or can be synthesized using different texture synthesis approaches. There have been many methods proposed for generating solid textures in the last decade due to the advances in graphics cards and the need for visualizing the interior of the objects. In Pietroni et al. (2010) these methods are divided into two categories:

- Shape-independent methods: These methods synthesize solid texture that are not specific to a model. They are assumed to be uniform inside the volume.

- Shape-dependent methods: These methods generate textures for the inside of a specific object. They are considered to be model-specific: they rely on the boundary information and guidance information specific to that model.

In this section I will present these methods and their application for medical visualization in detail.

2.3.4.1 Shape-independent solid textures

As in 2D texture synthesis, shape-independent methods can be categorized as procedural or exemplar-based:

1. Procedural approaches use functions of the 3D position of each voxel inside the volume to calculate its color value via a set of user-provided functions and their parameters. They are easy to implement and computationally efficient. A great advantage of this approach is that they do not use any memory for storing the texture. This makes them easy to use for visualizing high-resolution objects. However, finding the right function and its parameters for obtaining the desired texture can be difficult for non-expert users.

2. Exemplar-based solid texture synthesis denotes the process of constructing a large 3D volume texture from a small 2D or 3D sample texture by considering the content inside the sample. These methods can be categorized as follows:

   (a) Statistical methods: These methods require the computation of a specific statistical feature of a given texture and the synthesis of a solid texture that has statistical features similar to those in the input texture. In practice, input exemplar textures can be 2D or 3D. However, obtaining 3D textures is really hard in the real world, since the objects should be scanned in 3D using special equipment. Thus, in most of the recent work in this area the focus has been on using 2D exemplar textures as input. However, since the input is 2D and the output is a 3D texture, replicating the statistics becomes an issue in these methods. In Heeger & Bergen (1995) histogram matching to match the synthesized texture's color distribution with that of the input exemplar has been tried. Later, Jagnow et al. (2004) utilized stereology information for the textures that are composed by models of different particle shapes. Stereology provides 3D information about the given 3D data from measurements made

on 2D planar sections. The approach proposed by Jagnow et al. (2004) requires analysis of the distribution and shape of the particles in the 2D input image and uses it in solid texture synthesis. Though this approach produces good results, it is restricted to those solid textures that have particle shapes. In the method presented in Qin & Yang (2007) a volumetric texture is synthesized using Basic Gray Level Aura Matrices (BGLAMs), which capture the input texture's co-occurrence probability distributions of gray levels. The main drawback of this approach is that it only works for greyscale images. If the color channels are uncorrelated, this approach can be used for obtaining satisfactory results. However, in many color textures the color channels are correlated, and treating the channels separately causes visual artifacts in the synthesized textures.

(b) Neighborhood-matching methods: These methods are based on using the neighborhood information of each voxel in the 3D volume to characterize the voxel and finding a similar neighborhood in the exemplar for that voxel. The main issues in these methods are the representation of the 3D neighborhood around a voxel for doing comparison with the input texture and the orientation of the input textures for specifying the color at a voxel. The earliest work in this direction was presented by Wei (2003). The key idea in this method is to consider a 2D exemplar texture for each orthogonal direction (basically x, y and z directions) in the neighborhood search to find the best match. These neighborhoods are also used in local optimization to converge to the best color. The main problems in this approach are blurring in synthesized textures that have high structural information.

(c) Energy-optimization-based methods: Kopf et al. (2007) extended the method presented in Wei (2003) for synthesizing homogeneous solid textures from 2D exemplars by combining energy optimization and histogram matching

techniques. This method preserves the local statistics of the texture using an optimization scheme, and it preserves the global statistics using a histogram matching technique. Although this approach produces good results for isotropic cubic solid textures, it does not handle anisotropy and spatial variation of textures inside the model.

There are many advantages of shape-independent solid textures:

1. They can be used for rendering different surface models by embedding the model into the solid texture domain.

2. They can usually be generated automatically. They do not require any input from the user. For instance, exemplar-based techniques only need the user for specifying some of the parameters (e.g., neighborhood length, iteration count) for synthesizing textures.

3. They do not cause any distortion on the surface of the model since they don't consider the shape of the object.

The disadvantages of this approach can be summarized as follows:

1. They need high amount of memory for storage.

2. They do not consider any object-specific information.

3. They create solid textures which have the same anisotropy throughout the volume.

2.3.4.2 Shape-dependent solid textures

To synthesize shape-dependent (model-specific) solid textures, an appropriate representation (parameterization) of the interior of the object is required. The methods in this category can be used for synthesizing progressively-variant solid textures in order to illustrate objects using their model-specific material and shape properties as guidance.

These advantages can be used for obtaining a large variety of model-specific volumetric effects for many visualization and graphics applications.

1. Procedural approach: The method proposed in (Cutler et al. (2002)) is known as the first method to synthesize model-specific textures. This method uses a scripting language for specifying the regions inside the model and signed distance fields for computing these regions. It uses procedural methods for generating layer-specific textures on the cross sections.

2. Quasi-solid texture synthesis approach: These methods create 2D textures on different cross sections of the model.

   (a) Volumetric illustrations: The approach presented by Owada et al. (2004) provides a way to create quasi-solid textures for illustrating object interiors. This approach utilizes user input from a browsing interface and a modeling interface. The browsing interface enables the user to split the model as a means of specifying the interior cross section surface to be textured. The modeling interface is designed for specifying the internal regions on that specified cross section. Given the inputs that come from these interfaces, their framework synthesizes isotropic, layered, or oriented 2D textures on the cross sections. Though this method is an important approach for creating scientific illustrations, such as medical, biological, and geological, their expressive power is limited, and they require too much input from the user for our application. In addition, since 2D textures are synthesized on the cross sections every time the model is cut, it leads to inconsistency among different cross-sections.

   (b) Texturing internal surfaces from a few cross sections: Similar to the volumetric illustrations method in Owada et al. (2004), the proposed method in Pietroni et al. (2007) is based on creating quasi-solid textures in the interior of the model. The special property of this method is that it uses photographs

of cross sections of real objects to generate textures using a warping approach instead of a texturing approach. This method consists of a preprocessing and a morphing stage. In the preprocessing stage the images are placed on the model's local reference frame on the selected cross sections. In the morphing stage warping between these cross sections is used to generate textures on the remaining cross sections. This method is good for obtaining realistic volumetric illustrations of models that have highly structured textures. However, it cannot synthesize stochastic textures, and it requires the object domain to be closed.

3. Warp-based solid texture synthesis approach: Takayama et al. (2008b) presented a method that builds progressively changing oriented solid textures by repeatedly pasting 3D texture exemplars within a tetrahedral mesh. This approach uses a 3D exemplar texture for specifying the type of the output texture. It generates solid textures inside a model using the variation of textures, the amount of anisotropy, and the volumetric tensor fields, all obtained from user input. The synthesis process is based on pasting a patch in the object's interior volume. While this method creates volumetric textures for many models with little memory, it has some drawbacks. As input it requires 3D solid textures, which can be difficult to obtain for real objects. In addition, for highly structured textures the patch seams become noticeable in the synthesized texture.

These methods have many advantages for illustration purposes:

1. They provide user interaction that gives the ability to design different appearances for a single model.

2. They create progressively changing solid textures that follow an object's shape using the boundary constraints and user-specified volumetric regions.

There are also some weaknesses in the proposed approaches:

1. They require the boundary geometry to be well conditioned. For instance, Owada et al. (2004) and Pietroni et al. (2007) require the model to be closed.

2. They do not provide correspondence between different instances of the same model since they require user input for each model.

3. They may cause distortions of the textures due to the spatial parameterization of the models.

2.3.4.3 Hybrid methods

The shape-independent methods can be modified to synthesize shape-dependent textures. To achieve this, there are some suggestions in the existing literature. For instance, in Pietroni et al. (2010) a constrained synthesis approach using a 2D mask that approximates the surface boundary is recommended as a possible extension for adding the boundary dependency to the synthesized textures. In addition, it is suggested that layered textures can be synthesized from multiple sources according to a given mask using already existing methods.

Combining shape-independent and shape-dependent texture synthesis methods is a major focus of this dissertation. The details of the proposed approach are explained in Chapter 6.

2.3.4.4 Solid texture synthesis for medical visualization

Volume illustrations became popular over the past decade in medical visualization for their effectiveness in illustrating the material and structural features of a data set while emphasizing important details. One approach is to generate volumetric illustrations by synthesizing model-specific textures. Dong & Clapworthy (2005) proposed a method for synthesizing oriented volumetric solid textures via a patch-based solid texture synthesis algorithm that uses the orientation field extracted from CT and 3D exemplar textures

provided by the user.

Later, Lu & Ebert (2005) proposed a method for generating illustrative 3D textures using Wang Cubes. This method basically uses 3D texture samples from the available volume datasets as exemplars. First, the method recolors these exemplar textures to imitate the 2D illustrations in medical textbooks. Then, it generates output texture using a Wang Cubes approach. The main disadvantages of this method are that the textures do not reflect scale and orientation information, and they are not region-appropriate but rather are homogeneous throughout the volume. In addition, 3D texture samples are not easy to capture, and tiling of the 3D textures leads to repetitive patterns.

There are also some methods proposed for synthesizing oriented solid textures for visualizing muscle fibers. Chen et al. (2009) proposed a method for the visualization of muscle based on diffusion tensor images (DTI) using oriented solid textures. They utilized a vector field computed from the DTI data as a guide to synthesize solid textures.

2.4 Object Representations in 3D Space

In computer graphics there have been many methods proposed for representing objects in 3D space (Funkhouserl (2002)):

1. Raw data: Point cloud, range image, polygon soup
2. Surfaces: Mesh, subdivision, parametric, implicit
3. Solids: Voxels, binary space partitioning (BSP) tree, sweep
4. High level structures: Scene graph, skeleton, application-specific

As discussed earlier, depending on the type of texture synthesis, surface-based or volumetric-based, a representation of the boundary or interior of the model is needed. In this section backgrounds on surface representations and volume representations are

given first. Then, an overview of the model-based coordinate system utilized in this dissertation is presented.

2.4.1 Surface representations

Surface representations are used for defining the boundary of an object. There are many criteria in surface representation. It must be accurate and concise. It should have local support, and it should guarantee continuity. Methods trying to satisfy these goals have been proposed for representing surfaces:

1. Discrete representation (mesh): Meshes consist of a connected set of polygons (usually triangles). Complex objects can be represented easily using this method. Independent faces, vertex and face tables, adjacency lists, and triangle meshes are some examples of mesh data structures used in this representation.

2. Subdivision surface: This representation describes the surface with a coarse mesh and a subdivision rule. The refinement stops when the surface is locally smooth. They are simple for describing complex surfaces. They are relatively easy to implement. They guarantee continuity and provide local support. On the other hand, they do not represent object interiors and exteriors.

3. Parametric surface: This representation describes a surface by a parameter equation f with 2 parameters, u and v. In this representation it is easy to represent points on the surface. In addition, this representation makes it possible to describe complex shapes. The Bezier surface is one example of parametric surfaces.

4. Implicit surface: This representation describes the surface using the points satisfying an equation f. This means all points (x, y, z) on the surface satisfy $f(x, y, z) = 0$. Implicit functions can be defined using algebraics, blobby models, skeletons, procedures, samples, and variational methods. They are widely used in computer

graphics applications. The main advantage of this method is that it is possible to distinguish the interior and exterior of the objects. In addition, it is easy to compute intersections, unions, and differences between such surfaces. It is also easy to handle topological changes. However, it is difficult to define an arbitrary object with a function. It is hard to enumerate points on the surface, and it is hard to describe sharp features.

Among these methods, the mesh representation is the most common since it is easy to represent and manipulate complex objects in this representation. For many applications, such as texture mapping, resampling, surface painting, and 3D scanning, meshes need to be parameterized. Parameterization enables us to operate on the 3D surface as if it were flat. Given a triangle mesh, a surface can be parameterized using two parameters. Surface parameterization can be seen as a 1-1 mapping from the surface to a suitable domain. In general, surfaces cannot be flattened without some distortion. A good mapping is one which minimizes either angle distortions or area distortions. For surface parameterization there are many such mapping techniques: conformal mappings, harmonic mappings, and discrete harmonic mappings.

2.4.2 Volumetric (solid) representations

Surface representation techniques are not enough for various visualization and animation purposes since they do not capture the volumetric properties of the materials and they do not allow big geometric changes. In contrast, volumetric representations model the interior regions of the objects, which make them suitable for capturing the model's internal material structures.

2.4.2.1 Volumetric data structures

Different volumetric data structures have been proposed and used for different applications:

1. Voxels: In this structure, voxels, which are obtained by subdividing the space into tiny rectilinear solids, are arranged in a rectilinear grid to represent the volume. Each voxel can store material information such as color and opacity. One example of this data structure is medical data acquired from imaging devices, such as CT. This data can be stored at different pixel resolutions with different spacing between slices, and it can be rendered using volume rendering techniques. The advantages of this method are that voxel values can store any number of subvalues, and they can be accessed and manipulated easily. In addition, this representation has the same complexity for all objects. However, due to memory constraints it is hard to store high resolution volumes using this representation.

   Voxels can be displayed using isosurface rendering, slicing, and ray casting. Most of these rendering techniques are expensive to achieve. In addition, there are many problems that can be noticed in the rendering from voxels. For instance, there is the blurring problem due to averaging across each voxel. There is also the aliasing problem due to the discreteness of the representation and the sharp edges and corners in the voxel elements.

   An additional problem with voxels is that they do not contain shape relevant information.

2. Quadtrees and Octrees: Instead of using uniform sized voxels as in the voxel representation, here the resolution of the voxels is defined hierarchically. This representation is more concise and efficient for non-uniform representation of objects.

3. Slabs: This data structure is a surface-aligned volume data structure confined to a narrow region around the boundary of the object. In this method voxels are positioned and aligned locally with the surface. It has the advantage of combining a voxel-based representation with a surface-only representation (Dorsey et al. (1999), Agarwala (1999)). It has the advantage of converting distance functions from

$O(n^3)$ to $O(n^2)$ memory space. The thickness of the slabs can be specified. For thick slabs, more voxels are placed around the surface. The main advantages of this model are that it does not have scalability problem as in voxel-based models but it can still capture important volumetric information. However, for thin objects or objects with long skinny regions, this approach is not appropriate. In addition, alignment of the slabs to the surface is a complicated problem by itself. Note that the term slabs used by Dorsey et al. (1999), Agarwala (1999) refers to a different structure than what we used for slab in m-reps.

4. Tetrahedral Meshes: These meshes are generated from different surface models (e.g., meshes, medial models) or volumetric models (e.g., voxels) by decomposing them into a series of tetrahedra. They are mostly used for simulation purposes, such as finite element methods. The placement, size, and shape of the tetrahedra are the important aspects of this approach for obtaining physically plausible simulations. In Cutler et al. (2004), creating tetrahedral models of complex objects for the simulation and modeling purposes are investigated. In addition, in Takayama et al. (2008b) rendering of tetrahedral meshes with solid textures is presented. In this approach cubic solid textures are mapped on the interior of the model by assigning texture coordinates to each vertex of each tetrahedron inside the model.

2.4.2.2 User-guided volumetric modeling

The volumetric data structures summarized in Section 2.4 are used for representing the interior of the object. Assigning a value to each entity in a volume representation can be used to specify region fields (e.g., layers) or orientation fields inside a model. This can be used for visualization and simulation purposes.

Many user-guided approaches are proposed for specifying different regions inside the model:

1. Procedural approach: Cutler et al. (2002) presented a procedural approach for

generating layered solid models. In this approach, internal structures of the volume are specified by the user using a simple scripting language. This scripting language also provides sculpting and simulation operators for shaping and modifying the model.

2. Sketch-based approach: Owada et al. (2008) proposed a a sketch-based interface for designing volumetric data from scratch. In this approach voxels are painted by distributing colored particles along guided surfaces.

3. Diffusion surface approach: Takayama et al. (2010) proposed a representation called diffusion surfaces (DSs). This representation consists of 3D surfaces with colors defined on both sides. These colors inside the volume are computed by diffusing colors from nearby surfaces locally at user-defined cross sections using a modified version of the positive mean value coordinates algorithm. In their work they also present a simple sketch-based interface which enables the user to model objects with rotational symmetries, such as the inside of tomatoes. In this way, they model different objects which have smoothly changing global features inside the object.

There are also approaches proposed for assigning orientation information to each entity in the parameterized volume. For example, in (Takayama et al. (2008b)) an interface for designing volumetric tensor fields over tetrahedral meshes is proposed. This interface is used later for putting solid texture patches on the specified regions along the tensor fields.

2.4.3 Model-based coordinate systems

Model-based coordinate systems provide a parameterization based on the model, not based on world coordinates. Although there are other model-based coordinate systems, in this dissertation skeletal models with an unbranching medial curve or sheets are

considered since they yield a particularly nice along-, across-/around-, and through-object coordinate system.

A model-based representation method for computing the along-object (u), around-object (v), and through-object (τ) coordinates for any point in world space within each region inside the model in the scene is important for illustrating anatomical models. Consider an anatomical object, such as the duodenum or the sternocleidomastoid muscle (scm). For the duodenum these coordinates should progress from one end to the other and from the convex side to the concave side. For the scm these coordinates should progress from one end to the other and across the object. A model-based coordinate system can provide these two coordinate directions. Two surface-relevant coordinates (u, v) can be defined with respect to the model in this coordinate system. In addition, the τ coordinate, which tracks across the volume inside the 3D object, can be used for the computing the regions along the depth field inside organs for synthesizing region-specific solid textures, such as in bones (Figure 2.13).

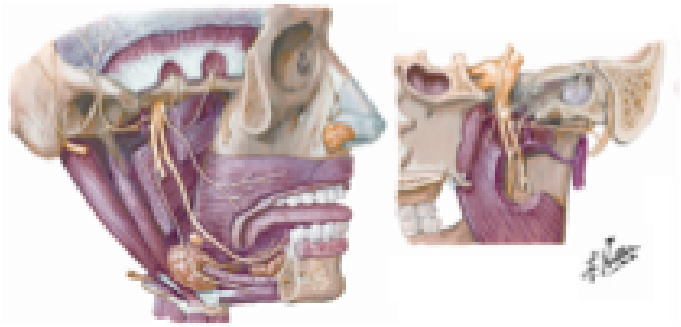

Figure 2.13: The material variation is along the depth field (through the object) for bones. (@2011 Elsevier)

These coordinates should also provide correspondence between different instances of the anatomical models in order to describe the relation between two instances of the same object. That is, corresponding points in different instances of an anatomic object, such as an anatomical structure in different patients, should have the same u,v,τ values. A model-based coordinate system, describing an object in an intrinsic model coordinate

system, is a way to achieve this goal.

2.4.3.1 Discrete medial representation

The medial representation provides the three desired model based coordinates (u,v,τ). For slabs like the scm, these are end-to-end, crest-to-crest-to-crest, and through-object. For tubular objects like the duodenum, these are along the tube, around the tube, and within the tube. The particular medial description system used in this dissertation is called an m-rep. M-reps (Pizer et al. (2003)) have been successfully used to represent anatomical objects and complexes of objects (Pizer et al. (2005), Chaney et al. (2004)), mostly in medical image segmentation and shape analysis.

An m-rep consists of one or more medial sheets or curves, and the part of the object associated with a particular sheet or curve is defined as a figure. So called "single figure slabular" m-reps (Siddiqi & Pizer (2008)) represent 3D objects by a single continuous medial sheet and a set of spokes. So called "single figure quasi-tubular" m-reps (Siddiqi & Pizer (2008)) represent 3D objects by a single continuous medial curve and a wheel of spokes across each point on the curve. The medial coordinate system is also well suited to represent an object with parts (Pizer et al. (2005), Chaney et al. (2004)), such as an object with a protrusion subfigure or an indentation subfigure (Han (2008)). In the m-rep of a multi-figure object each part is geometrically represented by a single figure m-rep, and they are connected by hinge geometry. In this dissertation the objects that are modeled using a single medial sheet or curve are considered for texturing purposes. But this approach could be extended to the multi-figure objects.

2.4.3.2 Single figure coordinate system

Medial geometry (Blum (1967); Siddiqi & Pizer (2008)) describes 3D objects in terms of a skeletal surface or axis, which lies midway between opposing surfaces of the object. Depending on the type of object, slab-shaped or tube-shaped, different parameterization

techniques are used for medial geometry:

1. Slab parameterization: In this parameterization medial geometry describes 3D objects in terms of a skeletal surface, a 2D curved sheet, and a set of spokes extending to the object boundary from both sides of the skeletal surface. The continuous medial manifold, M, of a 3D slabular object is a sheet of medial atoms parameterized by (u, v), the coordinates along the medial manifold, with u and v taking the atom index numbers for sample positions along and across M. The single entity $M(u, v)$ is called a medial atom. For slabular objects the medial atom is a modeling primitive that represents a track through the object interior (Fig. 2.14-left). Each medial atom has a hub, which is a point on the medial sheet $\underline{p}(u, v)$. From each hub two or three spokes extend from the medial sheet to the corresponding boundary points. To represent a hub, two spokes, and the common length of those spokes, the medial atom is defined as a 4-tuple $\{\underline{p}, r, \underline{U}^{+1}, \underline{U}^{-1}\}$ where $\underline{p}$ is the position vector, r is spoke length, and $\underline{U}^{+1}, \underline{U}^{-1}$ are the two spoke orientations as two unit vectors (Figure 2.14-left). The medial atoms on the edge of the medial sheet correspond to crests of the object boundary. Such an end atom adds a bisector spoke $\underline{U}^{0}$ and its length r_0 to the parameterization (Figure 2.14-right). In this parameterization moving in the along-object (u) direction refers to moving along the longer direction on the medial surface on both medial sides of spokes. Moving in the across-and-around-object (v) direction refers to moving along the shorter direction on the medial surface on one medial side and turning onto the other side at the crest (Figure 2.15).

2. Quasi-tube parameterization: In this parameterization medial geometry describes 3D objects in terms of a skeletal curve and a set of spokes extending to the object boundary from the sides of the skeletal curve. The continuous medial manifold, M, of a 3D tube-shaped object is a curve of wheel-like medial atoms parameterized by u, the coordinate along the medial curve. In this parameterization v moves

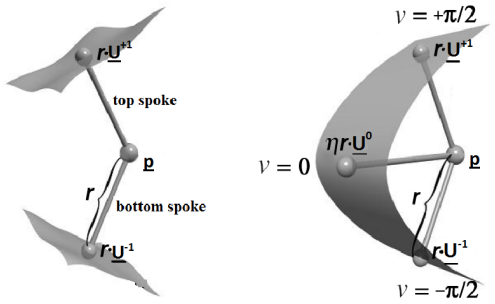

Figure 2.14: (Left) an internal medial atom; (right) a crest atom.

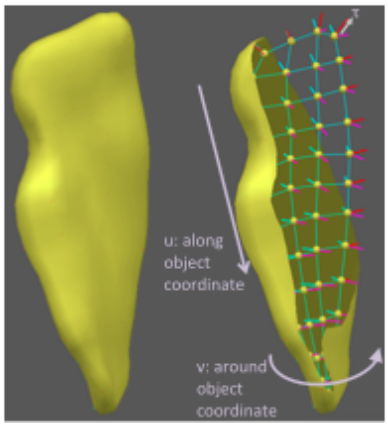

Figure 2.15: A single figure slabular m-rep for the scm and the object boundary implied by it.

cyclically around the wheel of spokes forming a cross section of the quasi-tube. The single entity $M(u)$ is called a medial atom. For tubular objects the medial atom is a modeling primitive that represents a plane of tracks through the object interior. Each medial atom has a hub, which is a point on the medial sheet $\underline{p}(u)$. From each hub spokes extend from the medial sheet to the corresponding boundary points. To represent a hub, spokes, and the common length of those spokes for a tube (quasi-tube in which every wheel of spokes is circular), the medial atom is defined as a 4-tuple $\{\underline{p}, r, \underline{U}^0\}$ where $\underline{p}$ is the position vector, r is spoke length, $\underline{U}^0$ is the orientation orthogonal to the wheel. For a quasi-tube each spoke in a wheel must have its own r value, and these values must be chosen such that each spoke ends in a common plane (Figure 2.16-right). In this parameterization moving in the along-object (u) direction refers to moving along the medial curve for all wheel

spokes, and moving in the around-object (v) direction refers to rotating around the wheels (Figure 2.17).

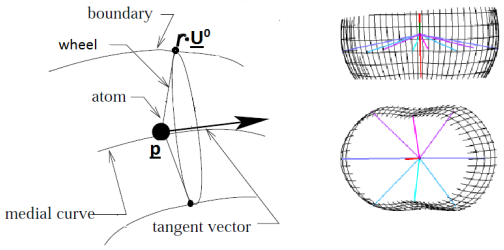

Figure 2.16: (Left) an internal medial atom for a tube; (right-top) a quasi-tube atom with spokes; (right-bottom) cut-away section of the surface.

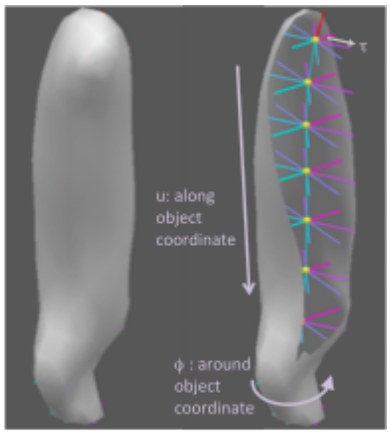

Figure 2.17: A single figure quasi-tubular m-rep for the pharynx and the object boundary implied by it.

The final coordinate, measuring distance along the spoke directions from the implied boundary is denoted by τ. It stores the fraction of the spoke from the medial sheet to the boundary. This coordinate can be used for distinguishing the inside regions of the figure from the outside. That is, $\tau < 1$ in the interior of the figure, $\tau > 1$ in the exterior of the figure, and $\tau = 1$ on the boundary.

Given an m-rep figure, a smooth object surface is generated to interpolate the boundary positions and normals implied by the atom spokes using either Thall's subdivision method (Thall (2002)) or Han's spoke interpolation method (Han (2008)) (the former was used in this dissertation). If (u, v) parametrizes the spokes emanating from the

medial sheet or curve, the implied boundary is parametrized by (u, v). Thereby, the coordinates on the medial sheet can be transferred to the boundary of the model. In the slab parameterization there are two corresponding points (u, v) on the boundary for every hub $\underline{p}(u, v)$ on the medial sheet. In the quasi-tube parameterization there are coplanar v points on the boundary for each hub $\underline{p}(u)$ on the medial curve. For two different instances of a model the correspondence is defined by the same medial coordinates for this model, which is also helpful to indicate the similar points in the instances of the model. In this way, m-reps provide positional, metric, and orientation correspondences (Fig. 2.18). In this dissertation correspondence is very important since we want consistent orientation fields and material transition fields for different instances of the same model.

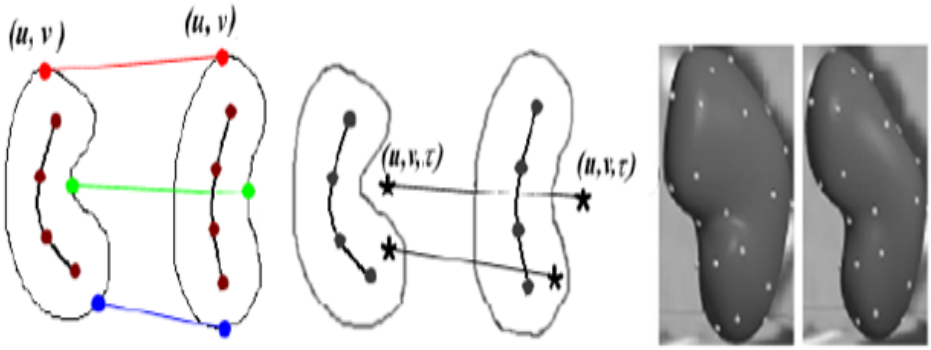

Figure 2.18: Medial-based correspondences between a figure and a deformed figure for boundary positions (left), and for interior and exterior positions to the boundary (middle). Boundary points on a deforming kidney (right).

Chapter 3

Texture Metamorphosis

In anatomical illustrations different regions of anatomical models are visualized using different material textures. The transition between these regions can be illustrated with textures that are perceptually between the two material textures.

To obtain this effect in the model-guided texture synthesis (MGTS) framework, region-specific exemplar textures along with the transition textures between them are needed as input. In this dissertation texture metamorphosis is used to create these transition textures by morphing one region-specific texture into another region-specific texture.

This chapter presents a technique that is accepted to the Computer Graphics Forum in 2011 as a journal publication (Kabul et al. (2011)).

3.1 Introduction

While image registration only deals with the alignment of a source image to a target image, image metamorphosis also considers changes in image appearance allowing the computation of transitions from a source image to a target image. Image metamorphosis has been of increasing importance in computer graphics as well as in medical imaging. It has been used to morph between faces, to mix multiple images, and to accommodate appearance changes in image registration in general.

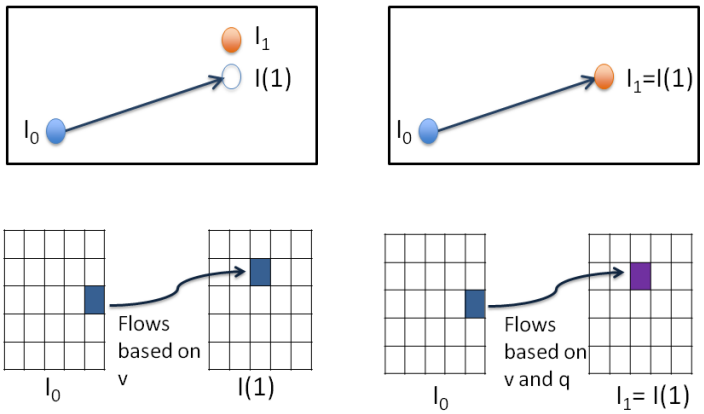

Figure 3.1: Comparison between registration (left) and metamorphosis (right). Here, the control variable v (the time and space dependent velocity field) controls the image deformation (flow of image over time), and the control variable q controls the change in appearance over time.

Image metamorphosis requires the computation of point correspondences (a time-dependent warp field) and an optimal change of pixel intensities over time. The change of pixel intensities (e.g., single intensities for a gray-value image, multiple intensities for color images) allows for an exact match to the target image. This joint optimization problem has for example been described in (Trouve & Younes (2005); Holm (2009)). Their method is a natural extension of fluid-flow registration methods that estimate time-dependent velocity fields to smoothly transform one image into another (Beg et al. (2005)). Figure 3.1 illustrates the difference between registration and metamorphosis. While image registration is considered to be an inexact matching problem, image metamorphosis is considered to be an exact matching problem.

Texture metamorphosis is a special case of image metamorphosis that is used to design new textures by interpolating between the input textures or to visualize the change of material on the objects by creating a smooth transition between the input textures. Structure features and appearance of the interpolated textures should perceptually be maintained between input texture-pairs. In this difficult problem, transformation of the intensity values as well as their feature masks, such as edges and ridges, should have an effect in the computation of the warping in order to split or merge structural features.

In this chapter a new texture metamorphosis approach is presented. This approach considers structural and appearance transitions of textures in a single framework. Consideration of structural features in an optimization framework is a novel approach. The proposed framework uses an optimal control framework (as proposed in Hart et al. (2009) for the image registration problem), and it uses a new numerical solution method that solves the metamorphosis problem as an initial value problem (and can therefore exploit a numerical solution strategy from a widely used fluid registration approach (Beg et al. (2005))). It is based on the large-displacement-diffeomorphic metric mapping (LDDMM) formulation of fluid registration (Beg et al. (2005); Hart et al. (2009)).

This new approach results in an intuitive solution method for texture metamorphosis that has a number of advantages (shown in Section 3.7) compared to other approaches for texture metamorphosis:

1. It is designed to interpolate textures while considering appearance and features in a single framework. This removes the need for an additional blending or synthesis scheme.

2. It can easily accommodate different image information, such as color or structural feature masks. Thereby structural information in the textures can have an effect on the deformation. The user can decide whether s/he would like to include that control in the deformation.

3. It can easily accommodate different measures of image/feature similarity, such as simple sum of squared differences, correlations, or mutual information.

4. It is designed to be symmetric with respect to the two textures.

5. It is robust and automatic. It requires neither user input for feature correspondences nor the selection of transition functions.

6. It interpolates structural textures as well as stochastic and natural textures.

The remainder of the chapter is organized as follows: Section 3.2 presents the overview of the basic image registration approach on which the metamorphosis is based. Section 3.3 presents the overview of the optimal-control-based texture metamorphosis method. Section 3.4 details this method for general textures. How to add color information is explained in Section 3.5. Section 3.6 details the texture metamorphosis approach for structural textures. Section 3.7 presents results.

3.2 Image Registration

Image registration involves finding a transformation that maps one image to another image as well as possible. In contrast to image metamorphosis, image registration is only concerned with determining this mapping and does not consider appearance changes. Therefore, perfect image matching is in most cases not achievable in image registration due to image noise, structural differences, or appearance differences in the images.

In fluid flow registration the deformation is achieved by flowing the image I_0 by a time-dependent velocity field v. The fluid flow registration problem is highly complex because the full spatio-temporal velocity field needs to be estimated. By choosing an appropriate norm to penalize non-smoothness, diffeomorphic image transformations can be assured.

The inspiration for the proposed metamorphosis approach comes from LDDMM fluid flow registration. In that formulation the energy equation to be minimized is (Beg et al. (2005); Hart et al. (2009)):

$$\begin{aligned} E(v) &= \int \|v\|_L^2 \, dt + \frac{1}{\sigma^2}\|I(1) - I_1\|^2, \\ \text{subject to} \quad & I_t + (DI)v = 0, \; I(0) = I_0 \end{aligned} \tag{3.1}$$

where I_0 is the initial image, I_1 is the target image, and σ is a scalar which controls the tradeoff between image match and transformation smoothness. See Hart et al. (2009)

for details on how to solve such an optimization problem.

3.3 Texture Metamorphosis

Given two exemplar texture images I_0 and I_1, the aim is to create interpolated images $I(t)$. Here, t corresponds to interpolation time where $t \in [0, 1]$ under the constraint that $I(0) = I_0$ and $I(1) = I_1$. This enforces that the initial and the final texture images are matched exactly under metamorphosis.

Since there are infinitely many possible ways to morph image I_0 into I_1, the solution space needs to be constrained in a meaningful way. In metamorphosis this is accomplished by formulating an optimization problem which seeks to determine a time- and space- dependent velocity field v *and* a time- and space- dependent source term (texture intensities) q which are appropriately regularized. This means it should seek a deformation and a texture variation both of which change smoothly across space at each t.

To allow exact matching of the texture appearance, the metamorphosis formulation in equation 3.2 adds the control variable q to the transport equation. This variable controls the image intensity change that is penalized according to $\|q\|_Q^2$.

In an optimal control setting the metamorphosis optimization problem is equivalent to minimizing the following energy E under a dynamic constraint for the image change:

$$E(v, q) = \int_0^1 \|v\|_L^2 + \|q\|_Q^2 \, dt, \tag{3.2}$$
$$\text{subject to} \quad I_t + (DI)v = q, \; I(0) = I_0, \; I(1) = I_1$$

where the control variables v (the time and space dependent velocity field) controls the image deformation (flow of image over time), q controls the change in appearance over time, and D denotes the Jacobian. The deformation map can be computed from v by integration. L and Q denote norms to penalize change in v and q, respectively. Here,

$\|v\|_L^2 = \langle Lv, Lv\rangle$ in which L is a differential operator to encourage smoothness of the velocity field over space, typically chosen of the form $L = -\alpha\nabla^2 + \gamma$. In this norm α encourages smoothness of the vector field (e.g., for large α strong deformations are discouraged, and the interpolation looks like blending); γ controls how much the overall travel distance of a particle is counted in the energy (e.g., a large γ will favor small overall displacements over an accurate image match). The norm for q is computed using $\|q\|_Q^2 = \langle Qq, Qq\rangle$, in which Q is a chosen linear operator. (For an easy numerical solution the operator is chosen as $Q = id$, the identity.)

The following constraints are imposed onto the optimization problem:

1. Image Flow: This constraint is simply a transport equation with a source term q; it controls the flow of the image from I_0 to $I(t)$. Intuitively it can be considered as a constraint which imposes that image intensity along a streamline of the velocity field v changes over time only through q. For $q = 0$ intensities will be constant along such a streamline (characteristic).

2. Initial and final constraints: To allow for texture morphing between two texture exemplars, the initial *and* the final condition for I are fixed. This is the main difference compared to standard image registration where an inexact match of the warped source image to the target image is admissible. This requires determining the time and spatially dependent q such that the exact match is achieved under the image flow constraint.

Note that the main difference to the registration problem in equation 3.2 is that the inexact matching term $\frac{1}{\sigma^2}\|I(1) - I_1\|^2$ has been replaced by the final constraint $I(1) = I_1$.

The challenging optimization problem in equation 3.2 is solved with the use of an adjoint solution method as proposed in Hart et al. (2009) for image registration. Details of the solution method and its relation to the corresponding image-to-image registration approach are given in the next section.

3.4 Solving the Texture Metamorphosis Equation

To solve this constrained optimization problem, it is converted to its unconstrained form by adding the dynamic constraint through a time- and space-varying Lagrange multiplier λ. Then the final state condition is added through the space-varying Lagrange multiplier τ. This makes minimizing equation 3.2 equivalent to minimizing the unconstrained energy:

$$E(v, q, \lambda, I, \tau) = \int_0^1 \|v\|_L^2 + \|q\|_Q^2 + \langle \lambda, I_t + (DI)v - q \rangle \, dt \\ + \langle \tau, I(1) - I_1 \rangle$$

with respect to $v(x,t)$, $q(x,t)$, I, and the Lagrange multipliers $\lambda(x,t)$, $\tau(x)$.

For a candidate minimizer of the unconstrained energy, its variation with respect to v, q, I, λ, and τ need to vanish. Computing the first variation and rearranging terms yields

$$\begin{aligned} \delta E(v, q, \lambda, I; dv, dq, d\lambda, dI) = \int_0^1 & \langle 2L^\dagger Lv + \lambda (DI)^T, dv \rangle \\ & + \langle 2Q^\dagger Qq - \lambda, dq \rangle + \langle I_t + (DI)v - q, d\lambda \rangle \\ & + \langle -\lambda_t - div(\lambda v), dI \rangle \, dt + \langle \lambda, dI \rangle |_0^1 \\ & + \langle d\tau, I(1) - I_1 \rangle + \langle \tau, dI(1) \rangle, \end{aligned}$$

assuming zero boundary conditions for v and a known $I(0) = I_0$. Since δE needs to vanish for any dv, dq, dI, $d\lambda$, and $d\tau$, fulfilling the boundary conditions (i.e., zero boundary conditions for v, $I(0) = I_0$, $I(1) = I_1$), at optimality the following conditions

need to hold:

$$\begin{aligned} I_t + (DI)v &= q, \quad I(0) = I_0, \\ -\lambda_t - div(\lambda v) &= 0, \quad \lambda(1) = -\tau, \\ 2L^\dagger Lv + \lambda(DI)^T &= 0, \\ 2q - \lambda &= 0, \\ I(1) - I_1 &= 0, \end{aligned} \tag{3.3}$$

These conditions can be categorized as follows:

1. Initial Condition: Sets the starting point for the images at time $t = 0$.

$$I(0) = I_0$$

2. State Equation (Transport Equation): Flows the image forward to time $t = 1$ using the velocity field v. The source term in the transport equation simply changes image appearance. This makes the change of intensity values in an image possible.

$$I_t + (DI)v = q$$

3. Final Conditions: Sets the final image and provides a final condition for the adjoint.

$$\begin{aligned} -\tau = \lambda(1) &= 2q(1) \\ I(1) &= I_1 \end{aligned} \tag{3.4}$$

4. Adjoint Equation (Scalar Conservation Law): Governs the dynamics of the adjoint (Lagrange mutliplier λ). Since the final condition $\lambda(1)$ is given by equation 3.4, these dynamics can be solved backward in time. The adjoint corresponds to the

concept of moments in physics (Miller et al. (2006)), and the scalar conservation law for the adjoint consequentially resembles the physical law of conservation of momentum.

$$-\lambda_t - div(\lambda v) = 0$$

5. Compatibility Condition: Computes the gradient of E and shows the relation between q and λ.

$$\begin{aligned} 2L^\dagger Lv + \lambda(DI)^T &= 0, \\ 2Q^\dagger Qq - \lambda &= 0 \end{aligned}$$

The first equation is used to compute the gradient of E with respect to v on the interval $t\epsilon[0,1]$. From $\delta E(v,q,\lambda,I;dv,dq,d\lambda,dI) = \int_0^1 \langle 2L^\dagger Lv + \lambda(DI)^T, dv\rangle dt$, the gradient $\nabla_v^{L_2} E(v) = 2L^\dagger Lv + (DI)^T\lambda$ with respect to the L_2 norm or $\nabla_v^V E(v) = 2v + (L^\dagger L)^{-1}((DI)^T\lambda)$ with respect to the V-norm (induced by the operator L) is obtained.

The second equation expresses that q is directly related to the adjoint λ (subject to a potential smoothing operator Q). To simplify the problem, it is assumed that the appearance control is penalized by an L_2 norm that makes $Q = id$, so the second compatibility condition becomes $2q - \lambda = 0$.

The optimality conditions reveal an interesting structure: the only differences from the image-to-image registration (Hart et al. (2009)) case are the source term q in the transport equation (the forward model), the additional compatibility condition on q, and the final condition on I.

3.4.1 Adjoint solution method

An analytic solution can in general not be obtained from the optimality conditions for metamorphosis. Numerical solution methods have been proposed in Miller & Younes (2001) and Garcin & Younes (2005). In both cases the metamorphosis problem is cast as a boundary value problem, subject to the fixed source and target images, and the solution proceeds by alternating gradient solution schemes for the image values (implicitly accounting for the image source terms, q, in the transport equation) and for the velocity field.

Here, by exploiting the relation between λ and q, an iterative update of the image I is used, and only gradient descent with respect to the velocity field v is performed. This approach results in an initial value problem that allows the use of a similar map-based solution as in Beg et al. (2005), which reveals the simple structure of the underlying optimality conditions and allows the integration of alternative image-matching terms (as they only affect the estimates for the velocity field through the final condition for the adjoint; see Hermosillo et al. (2002) for computations of these final conditions). While the use of maps is not strictly necessary for the solution of the optimization problem (Hart et al. (2009)), it is beneficial to avoid smoothing due to numerical dissipation effects.

Specifically, given an estimate of the velocity field v, the flow on I runs according to the forward deformation map $\Phi(t)$ computed from $t = 0$ to $t = 1$ using

$$\Phi(t) + (D\Phi)v \;=\; 0, \quad \Phi(0) = id, \tag{3.5}$$

and the reverse flow on λ runs according to the reverse deformation map $\Phi^r(t)$ computed from $t = 1$ to $t = 0$ using

$$-\Phi^r(t) - (D\Phi^r)v \;=\; 0, \quad \Phi^r(1) = id, \tag{3.6}$$

where id denotes the identity map, D denotes the Jacobian, and $\Phi^r(t)$ maps from $t = 1$ to $t = 0$. This allows the replacement of the direct solution for λ and I by solutions of transport equations on the maps Φ and Φ^r. The advantage of the map-based approach is that the maps are expected to be smooth (since the velocity fields are regularized according to L) and are therefore easier to propagate numerically.

In order to conserve the values of λ over time, the transformation is multiplied by $|D\Phi^r(t)|$, which corresponds to the amount of deformation the map has undergone:

$$\lambda(t) = |D\Phi^r(t)|\lambda(1) \circ \Phi^r(t). \tag{3.7}$$

Use of these maps to compute the solutions $I(t)$ is slightly more complicated than in the standard image registration problem (Beg et al. (2005); Hart et al. (2009)) due to the source term q, which needs to be integrated along the characteristics (streamlines) of the velocity field to affect the overall intensity value. This can be expressed by adding the change of intensity to the transformed image $I_0 \circ \Phi(t)$ as

$$\begin{aligned} I(t) = I_0 \circ \Phi(t) + \int_0^t q(\tau) \circ \Phi_{t,\tau}\, d\tau, \\ \text{where} \quad \Phi_{t,\tau} = \Phi(t) \circ \Phi^{-1}(\tau), \end{aligned} \tag{3.8}$$

which simply sums up all the q contributions over time in the current reference coordinate system at time t.

The proposed solution strategy builds on the observation that given a velocity field v, the final condition $\lambda(1)$ can be computed using the source term q that leads to the result $I(1) = I_1$ exactly. This is possible since we know from the optimality conditions that $q = \frac{\lambda}{2}$. Specifically, we observe that

$$q(1) = \frac{I_1^o - I(1)}{\int_0^1 \frac{1}{|D\Phi_{1,t}|}\, dt} \tag{3.9}$$

pointwise (the details of this equation are in the Appendix). Given $q(1)$, we can compute $\lambda(1)$ and in turn $q(t)$. Here, $I^o(1)$ denotes the solution to the source-less transport equation $I^o_t + (DI^o)v = 0;\ I(0) = I_0$ using the given velocity field v.

Algorithm 3.1 Texture metamorphosis

Input: I_0, I_1, L
Output: v
  Assume an intial flow field v is given
  repeat
    Flow image forward using
    $I^o_t + (DI^o)v = 0, \quad I(0) = I_0$
    Find $q(1)$ according to Eq. 3.9
    Find adjoint $\lambda(1)$ from $q(1)$ using $q = \frac{\lambda}{2}$
    Flow adjoint $\lambda(t)$ backward using Eq. 3.7
    Compute $q(t)$ from the solution of the adjoint using $q = \frac{\lambda}{2}$
    Compute the image $I(t)$ according to Eq. 3.8
    Compute the gradient for v from the compatibility conditions
    Perform a gradient descent step using $\nabla_v E$
  until convergence

The so-called adjoint solution method proceeds according to Algorithm 3.1, which is illustrated in Figure 3.1 and Figure 3.2. In effect, given an estimate of v, one (i) computes the solution to the source-less transport equation, (ii) uses it to compute the final condition for q, which then (iii) allows to the computation of $\lambda(t)$, $q(t)$, and $I(t)$, from which (iv) the gradient $\nabla_{v(t)} E$ can be computed. After taking a gradient descent step with respect to the velocity field, these steps are repeated until convergence. Figure 3.2 shows the intermediate results along with the deformation fields in each step.

Figure 3.3 illustrates the metamorphosis results. Here, the textures contain structural patterns in them. Unlike in linear interpolation, splitting and merging of these structural patterns should not lead to blurry textures in the interpolation. Using the proposed metamorphosis approach, even though there is no control in the structural features of these textures in metamorphosis, the structural patterns are split and merged only using the q term.

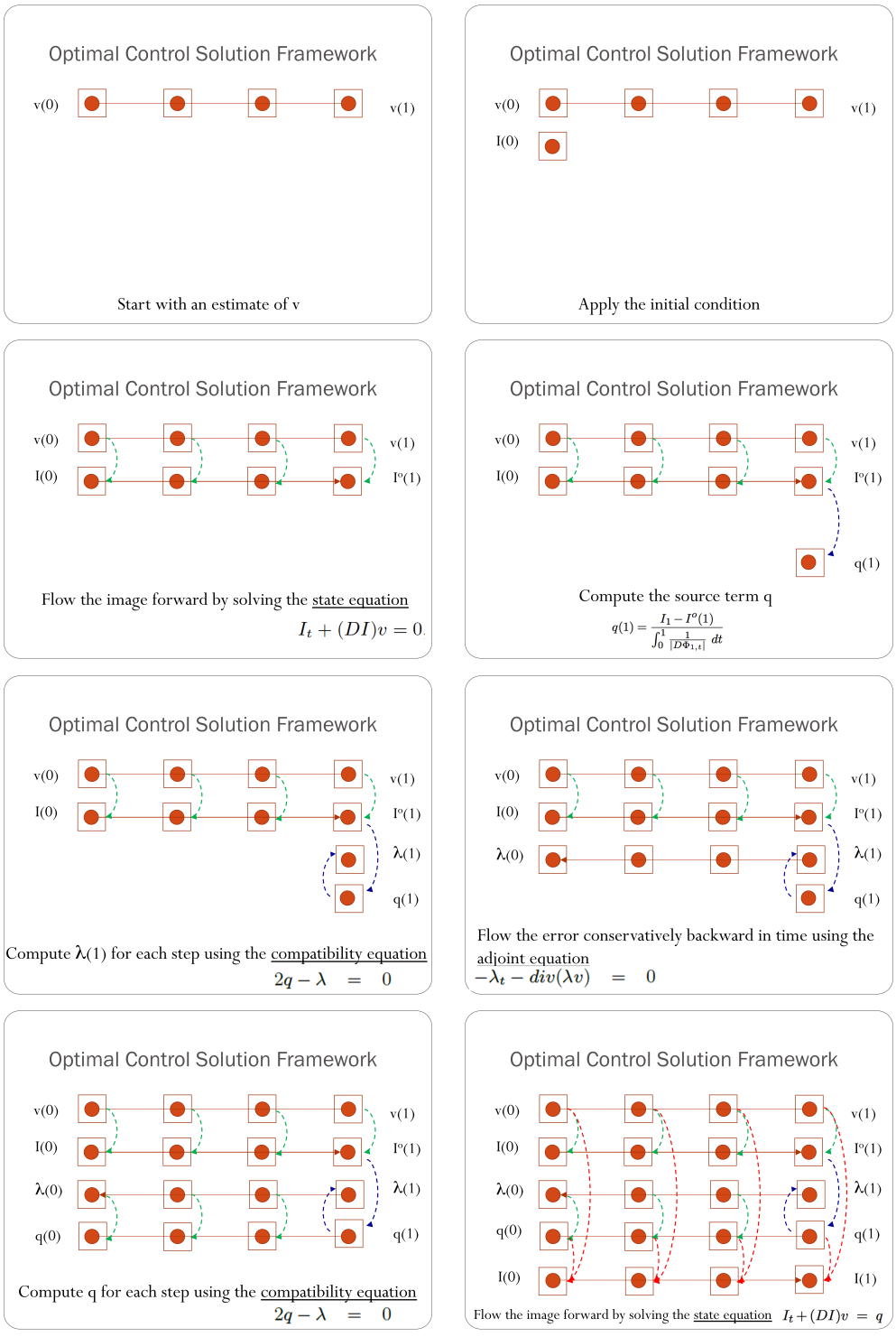

Table 3.1: Steps of the optimal control solution framework for computing $I(t)$.

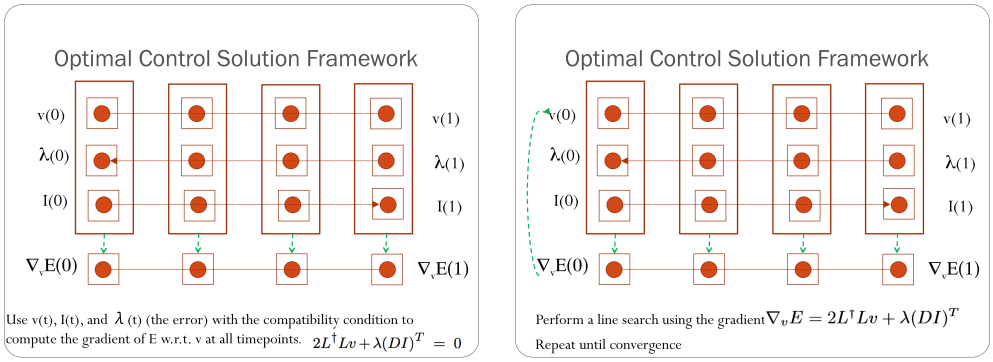

Table 3.2: Steps of the optimal control solution framework for computing $\nabla_v E$.

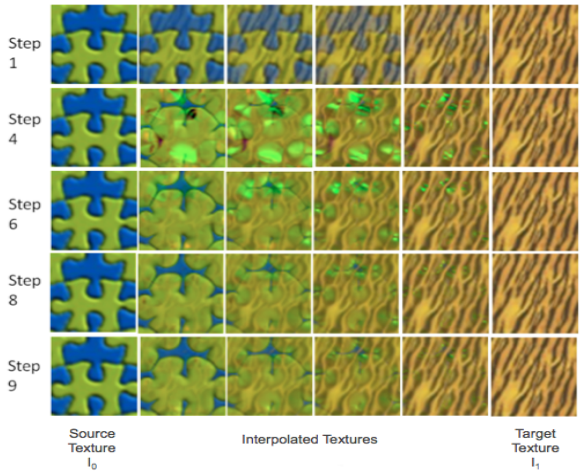

Figure 3.2: The texture sequences obtained in different steps of iteration. The rightmost image shows I_0, and the leftmost image shows I_1.

3.4.2 Combining forward and backward transitions in metamorphosis

According to the constraint $I(1) = I_1$, the morphed image $I(1)$ should always match exactly the target image I_1. However, due to numerical inaccuracies, the constraint may not be satisfied exactly. In order to avoid this problem, the whole solution method described in Section 3.4.1 is applied in the forward direction and in the backward direction, which allows us to make the problem symmetric and to obtain exact matching by construction. Specifically, two linearly weighted appearance control terms q_F and q_B are used in the energy equation. Here q_F controls the change in appearance from I_0 to I_1,

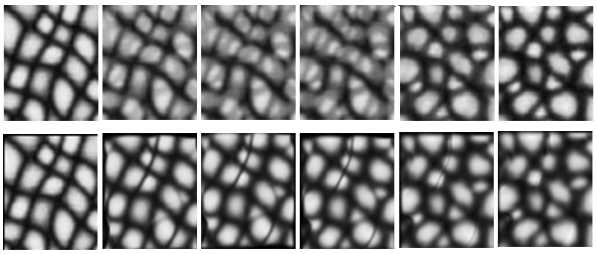

Figure 3.3: Metamorphosis from one texture to another using image intensities. The top row shows the results of linear interpolation, and the bottom row shows our results. The results obtained using linear interpolation are more blurry than the ones obtained using the proposed metamorphosis approach. As is seen, the end image in the top row is not exactly the same as the end image in the bottom row. This is due to the numerical inaccuracies; this problem is solved using the approach in Section 3.4.2.

and q_B controls the change in appearance from I_1 to I_0. The modified energy function is

$$\begin{aligned} E(v,q) = \int_0^1 \|v\|_V^2 + (1-t)\|q_F\|_Q^2 + t\|q_B\|_Q^2 \; dt, \\ \text{subject to} \quad I_{F,t} + (DI_F)v = q_F, \\ I_{B,t} + (DI_B)v = q_B, \\ I_F(0) = I_B(0) = I_0, \; I_F(1) = I_B(1) = I_1 \end{aligned} \tag{3.10}$$

where $I_F(t)$ denotes the transformed image in the forward direction from I_0 to I_1 at time step t, and $I_B(t)$ refers to the transformed image in the backward direction from I_1 to I_0 at time step t.

The resulting optimality conditions are similar to the ones obtained previously (now for two sets of equations) except for the compatibility condition, which becomes

$$2L^\dagger Lv + (1-t)\lambda_F(DI_F)^T + t\lambda_B(DI_B)^T \;\; = \;\; 0. \tag{3.11}$$

This simply amounts to solving the problem twice (in opposite directions) while keeping only one velocity field that needs to be estimated. The gradient for the velocity

field is then a linear combination of the gradients for the image solutions in the different directions. The image itself is recovered as

$$I(t) = (1-t)I_F(t) + tI_B(t).$$

In what follows it is assumed that all equations are solved numerically in this way without explicitly writing down the resulting forward/backward equations.

3.5 Metamorphosis for Color Textures

In texture metamorphosis the traditional approach is to warp single intensity values which come either from the texture image or its feature mask. One of the advantages of the proposed approach in this dissertation is that it can warp multiple color intensity values and structure features using a single framework. In this section addition of multiple color channels to the framework is explained.

The proposed framework allows multiple channels to be easily integrated into the energy equation by adding q terms for each channel. In that case the energy equation becomes

$$\begin{aligned} E(v,q) &= \int_0^1 \|v\|_L^2 + \sum_{i=1}^{n} \|q^i\|_Q^2 \, dt, \\ \text{s.t.} \quad & I_t^i + (DI^i)v = q^i, \; I^i(0) = I_0^i, \; I^i(1) = I_1^i \end{aligned} \tag{3.12}$$

where n is the number of channels that should be considered in the optimization framework. In this dissertation the red, green, and blue color channels are used. For each channel the same optimality conditions for metamorphosis are obtained independently (i.e., initial and final condition, transport equation, conservation law). The compatibil-

ity condition for v which is used to compute the gradient changes becomes

$$2L^{\dagger}Lv + \sum_{i=1}^{n} \lambda^i (DI^i)^T \quad = \quad 0. \tag{3.13}$$

3.6 Metamorphosis for Structured Textures

Some of the textures have strong structural components, such as edges, ridges, etc. in the appearance (intensities). These features model texture as a layout of regions. For texture metamorphosis, regions obtained using binary masking and signed distance fields to the edges are important to represent these layouts (Figure 3.4). These features F should be considered in the metamorphosis in order to obtain visually pleasing results especially since the structural components may undergo topological changes such as splitting and merging.

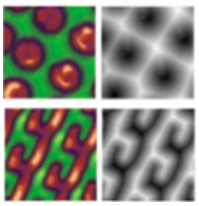

Figure 3.4: Texture features for metamorphosis. (Right) Textures; (left) textures' features obtained by computing signed distance fields to the edges.

In this optimization framework the aim is to simultaneously morph one image into another with respect to appearance while aligning their feature masks (without considering appearance change). This can be accomplished by adding the feature mask as an inexact matching term (as in image registration) subject to a transport equation *without* a source term, and also not having a final state constraint. The energy equation to be

minimized becomes

$$E(v) = \int_0^1 \|v\|_L^2 + \|q\|_Q^2 \; dt + \frac{1}{\sigma^2}\|F(1) - F_1\|^2, \tag{3.14}$$
$$\text{s.t.} \quad I_t + (DI)v = q, \; I(0) = I_0, \; I(1) = I_1,$$
$$F_t + (DF)v = 0, \; F(0) = F_0.$$

This can be combined with multiple color channels for metamorphosis as desired. Note that the feature image does not have a final state constraint. This makes the problem nonsymmetric. If desired, $F(0)$ could be estimated jointly such that $\frac{1}{\sigma^2}\|F(0) - F_0\|^2$ (as described in Hart et al. (2009)), which would result in a symmetric solution.

Adding the inexact matching term results in optimality conditions for F and its adjoint ω, which are the same as for image registration (Hart et al. (2009)). The compatibility condition for v changes respectively to include the influence of the feature image (i.e., the feature image influence is added). All other optimality conditions remain the same as in Section 3.4:

$$\begin{aligned} F_t + (DF)v &= 0, \quad F(0) = F_0, \\ -\omega_t - div(\omega v) &= 0, \\ 2L^\dagger Lv + \lambda(DI)^T + \omega(DF)^T &= 0, \\ F(0) - F_0 &= = 0, \\ \omega(1) &= \frac{2}{\sigma^2}(F(1) - F_1). \end{aligned} \tag{3.15}$$

See Algorithm 3.2 for a description of the solution steps.

In Figure 3.5 the comparison of the proposed approach with a texture-synthesis-based interpolation approach is shown. As can be seen in this figure, the intensity changes in the interpolated images are not smooth in the results obtained using the approach of Ray et al. (2009). In the results obtained using Ray et al. (2009)'s approach, the bright

Algorithm 3.2 Feature-based texture metamorphosis

Input: I_0, I_1,F_0, F_1, L
Output: v

Assume an initial flow field v is given
repeat
Flow image forward using
$I_t^o + (DI^o)v = 0, \quad I^o(0) = I_0$
Flow feature image forward using
$F_t + (FI)v = 0, \quad F(0) = F_0$
Find $q(1)$ according to Eq. 3.9
Find adjoint $\lambda(1)$ from $q(1)$ using $q = \frac{\lambda}{2}$
Flow adjoint $\lambda(t)$ backward using Eq. 3.7
Flow adjoint $\omega(t)$ backward using
$-\omega_t - div(\omega v) = 0$, $\omega(1) = \frac{2}{\sigma^2}(F(1) - F_1)$
Compute $q(t)$ from the solution of the adjoint
Compute the image I(t) according to Eq. 3.8
Compute the gradient for v from the compatibility conditions
Perform a gradient descent step using the gradients $\nabla_v E$
until convergence

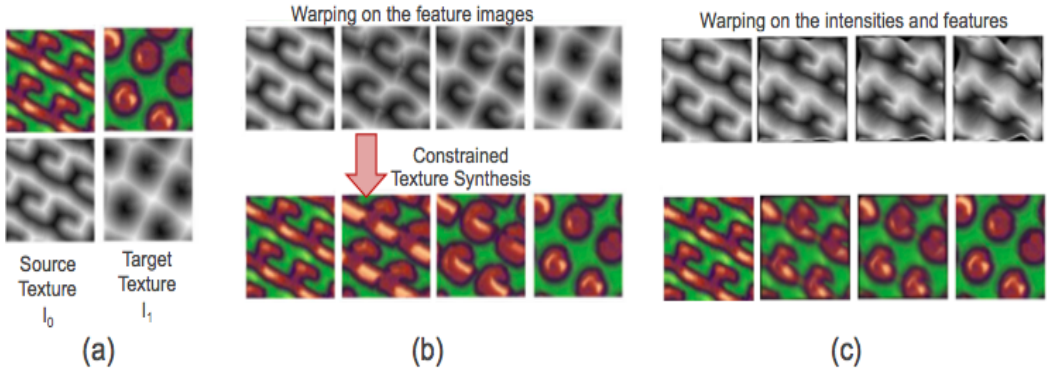

Figure 3.5: (a) Source and target textures with their feature images (signed distance fields to the edges). (b) The first and second rows show the results obtained using the approach in Ray et al. (2009). (c) The first and second rows illustrate our results. In each pair, even numbered rows show the texture images, and odd numbered rows show their feature image.

regions inside the pattern get bigger first, and then they get smaller to match with the target image. On the other hand, in the results of the proposed approach the bright regions move smoothly inside the patterns to generate the regions in the target image, even though the final feature images do not match.

3.7 Results of Texture Metamorphosis Variants

The proposed technique is suited for regular, semi-regular, and stochastic textures, since the deformation and appearance change is performed in a single framework.

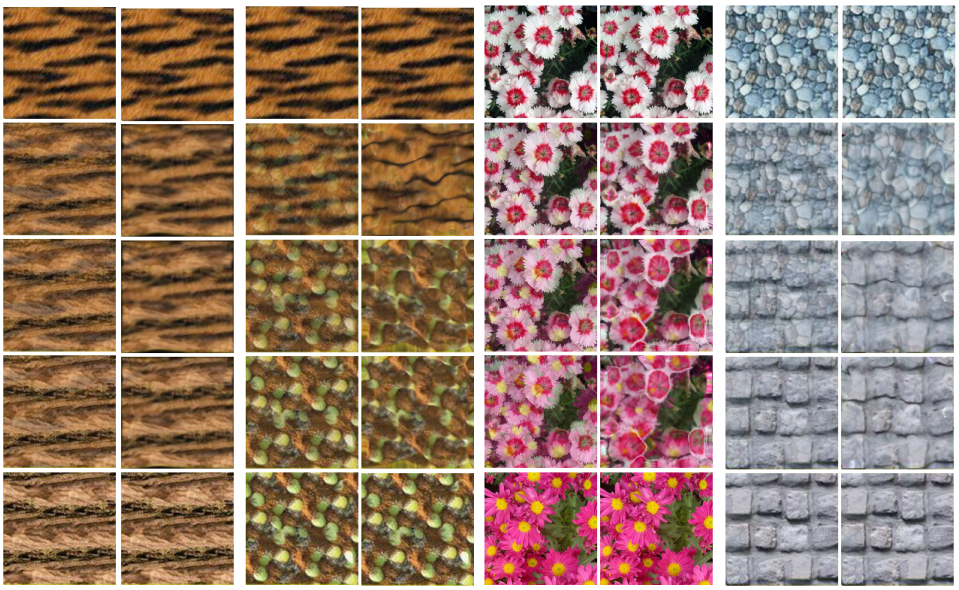

Figure 3.6: When meaningful features cannot be defined in the textures, such as natural textures, our metamorphosis can morph between textures by only considering appearance information. For each of the four texture pairs, the first column shows the results with linear interpolation, and the second column illustrates the results with our approach.

In Figure 3.6 the results generated using only image intensity information is presented. As can be seen, the approach created textures which are sharp and detailed, even when it is hard to define structures and the correspondences between them. Especially, the interpolation between pink flowers to white flowers created new flowers that have features present in both input images. The interpolated textures in this figure can

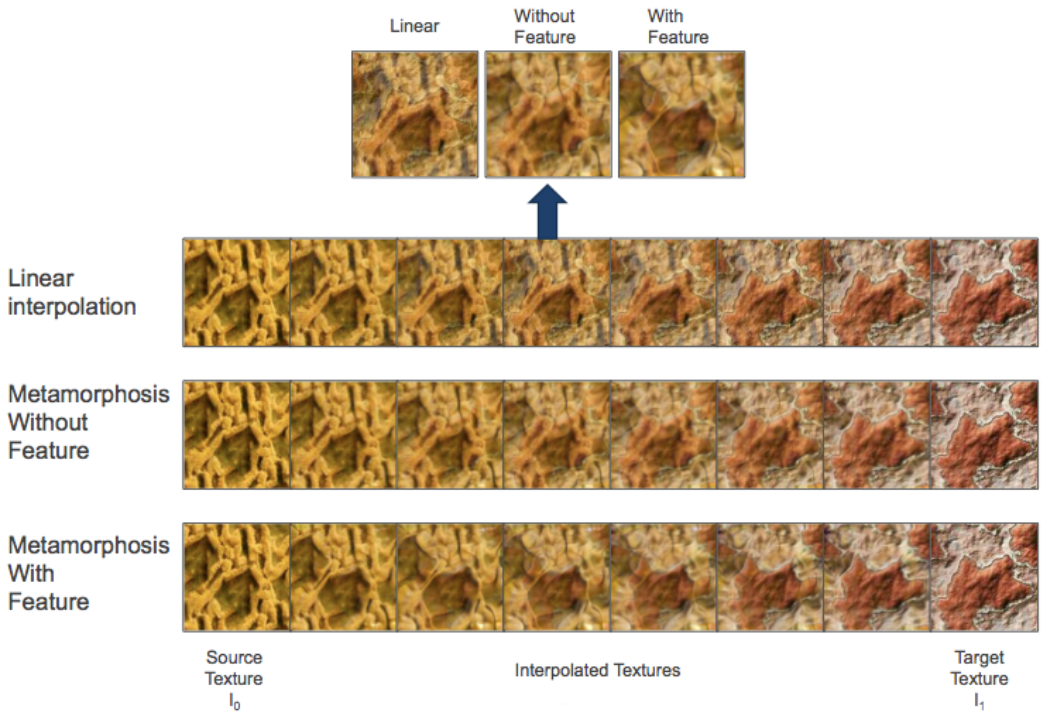

Figure 3.7: Including a feature channel in metamorphosis changes the deformation. (Top row) Linear interpolation; (middle row) metamorphosis without using structural features; (bottom row) metamorphosis using structural features.

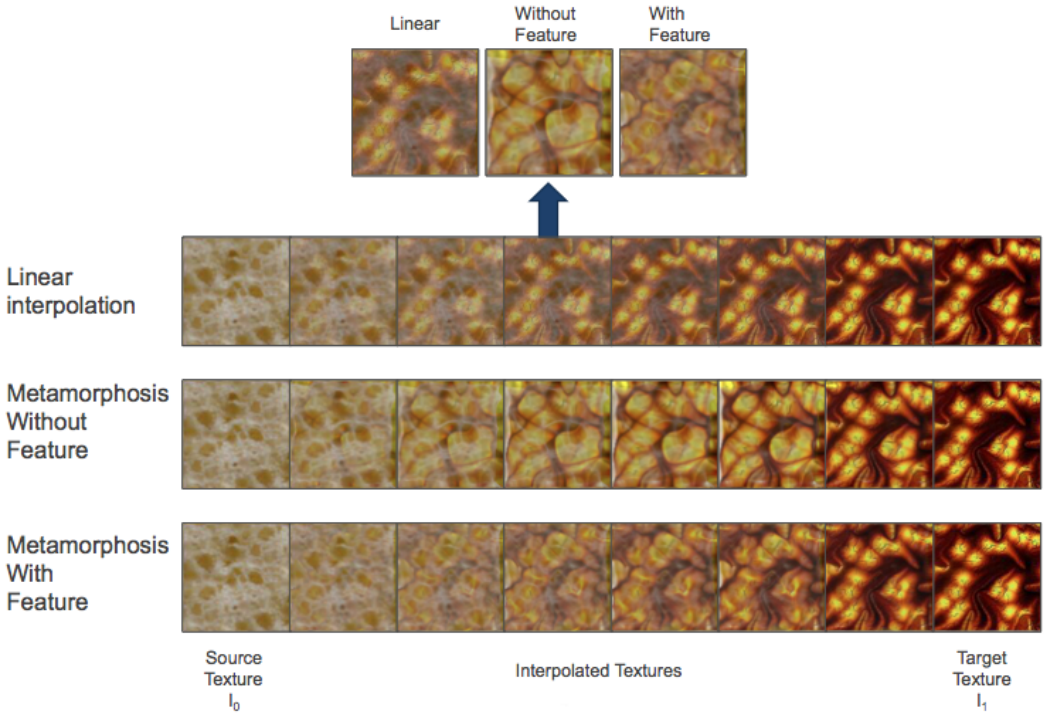

Figure 3.8: Including a feature channel in metamorphosis changes the deformation. (Top row) Linear interpolation; (middle row) metamorphosis without using structural features; (bottom row) metamorphosis using structural features.

be easily used in texture synthesis to generate textures on the surface of 3D models in order to visualize the change of materials over regions. On the other hand, if feature information is available, smooth and seamless transitions can be obtained. Figure 3.7 and Figure 3.8 show that the addition of feature channels in metamorphosis provides more control over deformation.

In Figure 3.9 and Figure 3.10 the comparison of the results with those presented in Ray et al. (2009) are shown. In Figure 3.9 since there are no meaningful structural patterns in the input textures, only image information in the energy equation is used. The proposed approach created smooth transformations from the leopard texture to the dice texture by forming the dots on the dice from the spots on the leopard texture.

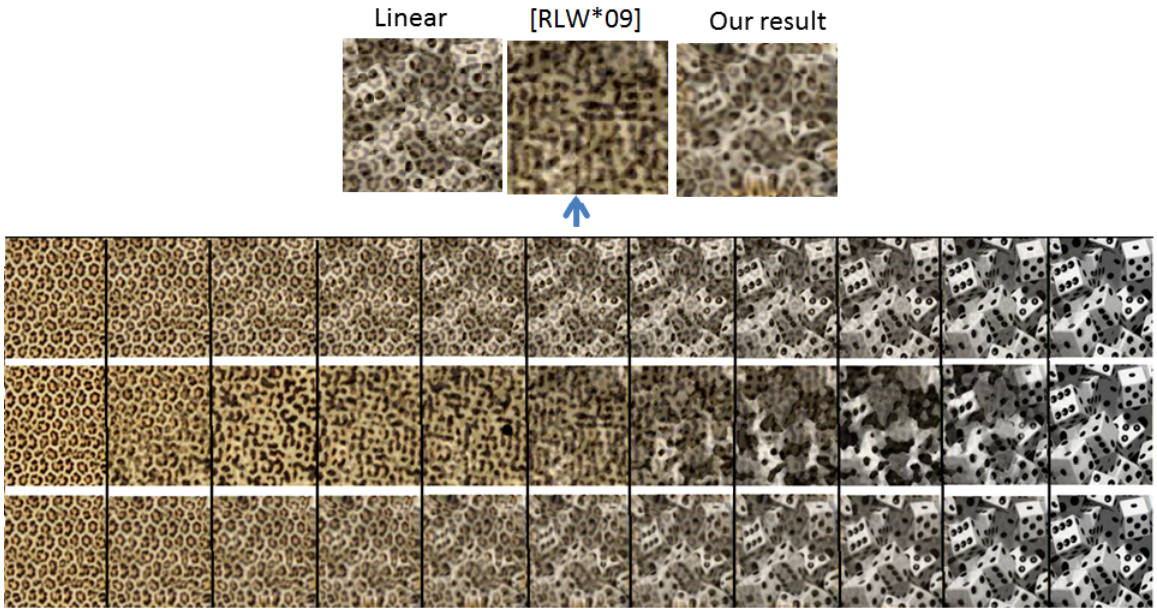

Figure 3.9: Comparison of interpolation results from Ray et al. (2009) with our results. The top row shows the results obtained using linear interpolation, the middle row illustrates the results from Ray et al. (2009), and the bottom row shows the results obtained using our approach.

In Figure 3.10 the structural feature information along with the image information is used to create interpolated textures. Since the method presented in Ray et al. (2009) interpolates texton masks using an advection algorithm and then synthesizes interpo-

lated textures constrained by these masks, it does not provide control over appearance transitions along with the texton masks.

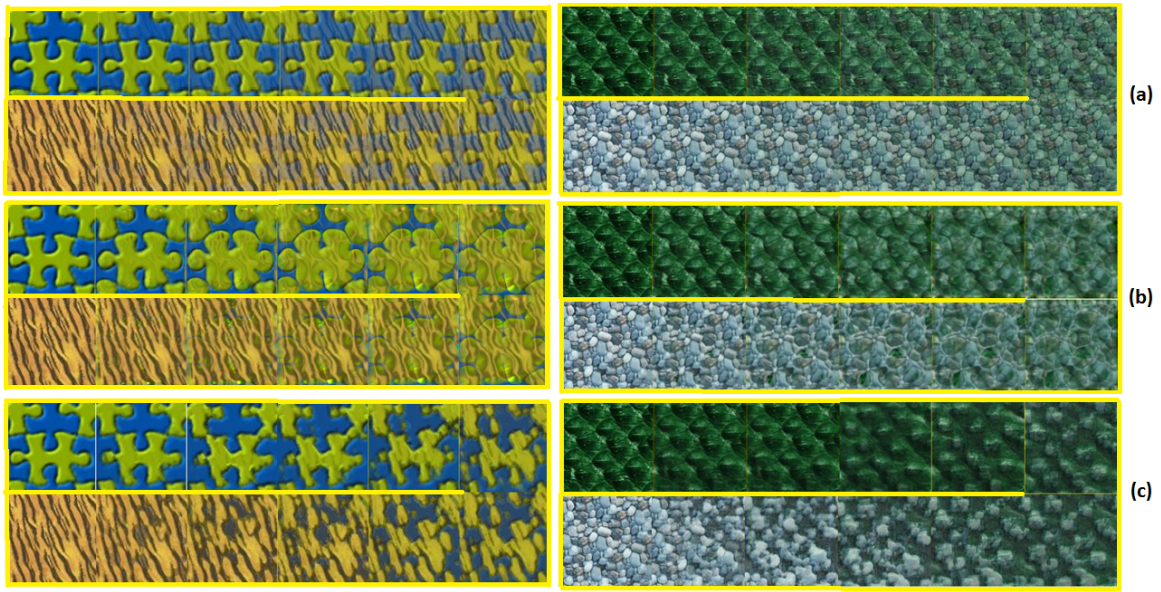

Figure 3.10: Comparison of interpolation results from Ray et al. (2009) with our results. (a) Linear blending; (b) our approach; (c) results from Ray et al. (2009).

On the other hand, the proposed approach in this dissertation deforms the structural features in the texture along with their intensity values, thereby providing control over intensity changes along with the structural features.

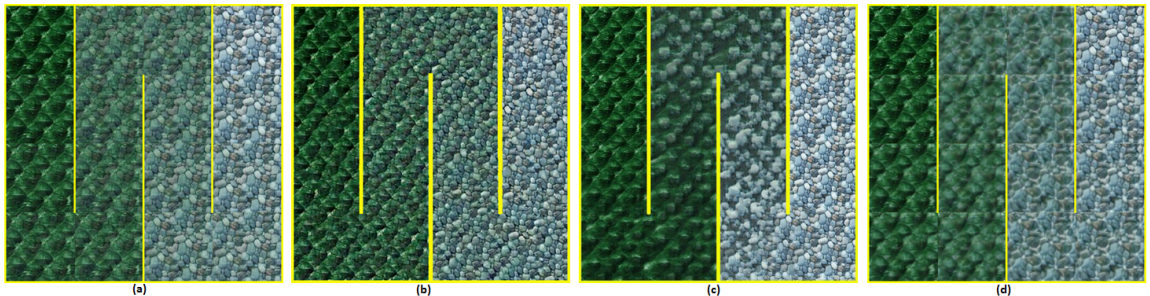

Figure 3.11: Comparison of results. (a) Linear blending; (b) results from (Ruiters et al. (2010)); (c) results from (Ray et al. (2009)); (d) our approach. Read each image sequence in column-major order; the upper left texture corresponds to I_0, and the bottom right texture corresponds to I_1.

In Figure 3.11 the results of the proposed approach are compared with those presented in Ray et al. (2009) and Ruiters et al. (2010). In this comparison, the textures

in the middle of the sequence are different from each other due to the fact that in each method the computation of deformation fields considers different information.

In Figure 3.12 the results are compared with the image registration method proposed in Hart et al. (2009). First, registration from I_0 to I_1 and I_1 to I_0 is applied. Then, the results obtained from them are combined using linear interpolation. As shown in the figures, the results obtained using registration and interpolation look more blurry.

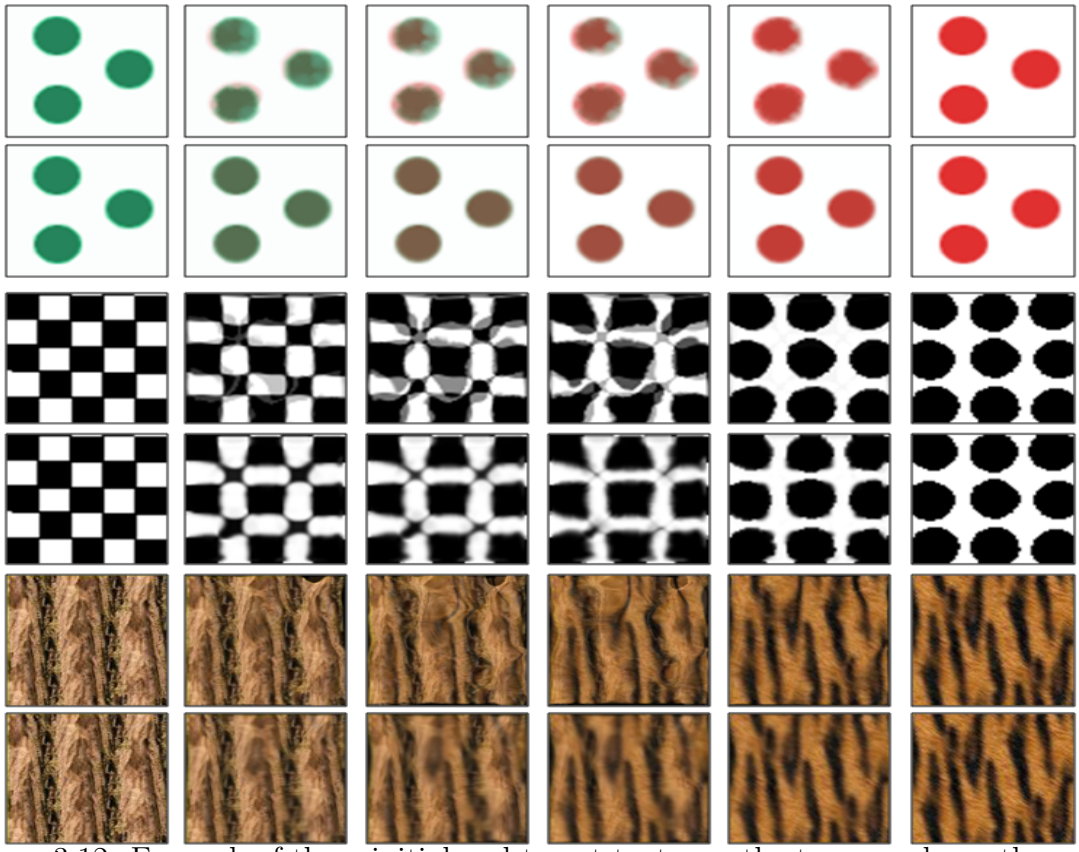

Figure 3.12: For each of three initial and target textures, the top row shows the results obtained using registration and linear interpolation for each channel, and the bottom row shows the results obtained using our metamorphosis approach. In the first row due to color values of green and red, which are (0,1,0) and (1,0,0) respectively, the dots are deformed by warping.

In Figure 3.13 medical applications of texture metamorphosis are illustrated with textures of the prostate. Interpolated textures are obtained from the boundary regions to the inner regions using the proposed metamorphosis approach. These textures can be used as exemplar textures for creating model-specific solid prostate textures.

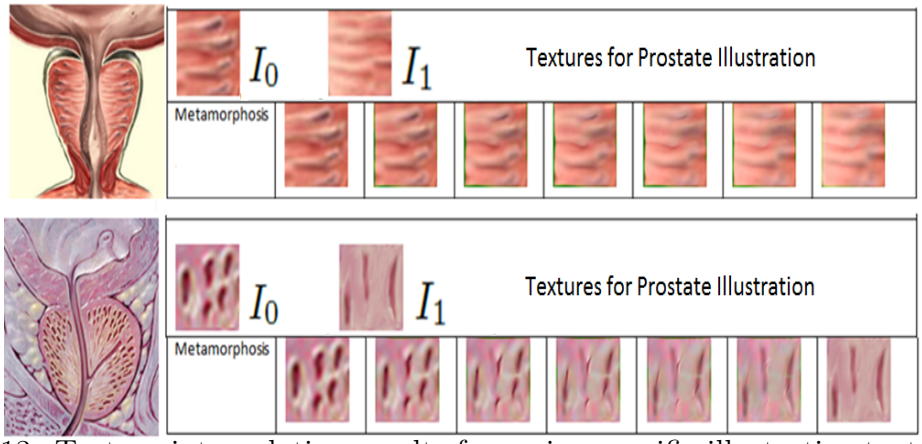

Figure 3.13: Texture interpolation results for region-specific illustrative textures of the prostate.

In Figure 3.14 the effect of changing α in the differential operator L is illustrated. Here, it can be seen that for large α strong deformations are discouraged, and the transformation becomes like blending.

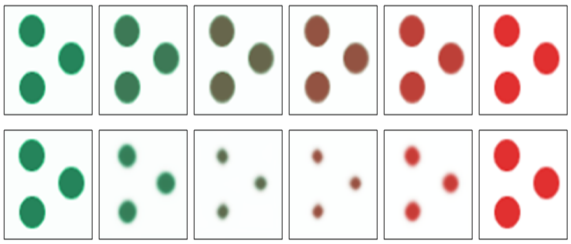
Figure 3.14: Top row illustrates results obtained by setting α to 0.01, and bottom row illustrates results obtained by setting α to 0.001.

3.8 Discussion

In this dissertation the metamorphosis problem is treated as an initial value problem. This property allows integrating additional terms into the framework easily with respect to the formulation as a boundary value problem. For instance, the proposed framework can easily accommodate different measures of image/feature similarity, such as simple sum of squared differences, correlations, or mutual information. This feature can be used to have more control on the features of the textures. For instance, if the user wants to have control of the splitting and merging of the regions, similarity between landmarks

and their order can be integrated into the energy term. The other limitation of the proposed approach is that it only considers two input textures as input and warps one to the other one. It would be interesting to design a new optimization framework with additional penalty terms and constraints to perform the interpolation among a number of input textures. This can be used for interpolating multiple textures.

3.9 Appendix

3.9.1 Optimality conditions

For a minimizer of energy equation 3.3, its variation with respect to v, λ, I, and γ need to vanish. Computing

$$\delta E(v, q, \lambda, I; dv, dq, d\lambda, dI) = \frac{\partial}{\partial \epsilon} E(v + \epsilon dv, I + \epsilon dI, \lambda + \epsilon d\lambda) \mid_{\epsilon=0}$$

yields

$$\begin{aligned}\delta E(v, q, \lambda, I; dv, dq, d\lambda, dI) = \int_0^1 & \langle 2L^\dagger Lv, dv\rangle \\ & + \langle 2Q^\dagger Qq, dq\rangle + \langle d\lambda, I_t + (DI)v - q\rangle \\ & + \langle \lambda, (dI)_t + (DdI)v + (DI)dv - dq\rangle \; dt \\ & + \langle d\tau, I(1) - I_1\rangle + \langle \tau, dI(1)\rangle\end{aligned}$$

since

$$\int_0^1 \langle \lambda, dI_t\rangle dt = \int_0^1 \langle -\lambda_t, dI\rangle dt + \langle \lambda, dI\rangle|_0^1,$$

and (by Green's theorem)

$$\begin{aligned}\langle \lambda, (DdI)v \rangle &= \langle -div(\lambda v), dI \rangle + \int_{\partial\Omega} dI \lambda v \cdot dS \\ &= \langle -div(\lambda v), dI \rangle.\end{aligned}$$

By assuming v vanishes on the domain boundary, we get

$$\begin{aligned}\delta E(v, q, \lambda, I; dv, dq, d\lambda, dI) = \int_0^1 \langle 2L^\dagger Lv \\ +\lambda (DI)^T, dv \rangle + \langle 2Q^\dagger Qq - \lambda, dq \rangle + \langle I_t + (DI)v - q, d\lambda \rangle \\ +\langle -\lambda_t - div(\lambda v), dI \rangle \; dt + \langle \lambda, dI \rangle |_0^1 \\ +\langle d\tau, I(1) - I_1 \rangle + \langle \tau, dI(1) \rangle,\end{aligned}$$

assuming zero boundary conditions for v and a known $I(0) = I_0$. Since δE needs to vanish for any dv, dI, $d\lambda$, fulfilling the boundary conditions of the problem, the optimality conditions are obtained.

3.9.2 Source term

To be able to compute the values for q without implicitly obtaining them through a gradient descent on I, we can make use of the fact that, for the L_2 norm penalizer, q is directly related to the adjoint λ and therefore also has to fulfill a scalar conservation law.

For such a transport equation without a source term, values stay constant along the characteristics (streamlines) of the transport equation. With a source term the source values get integrated along the characteristic. Hence, along a characteristic

$$I(t) = I(0) + \int_0^t q(t) \; dt$$

needs to hold. In particular, for our exact matching problem we need to have

$$I_1 - I_0 = \int_0^1 q(t)\ dt.$$

However, representing everything in the coordinate frame of image I_1 we know that

$$q(t) = \frac{1}{|D\Phi_{1,t}|} q(1)$$

since it needs to fulfill a scalar conservation law. Assuming we discretize time into n intervals spanning $[0, 1]$ and assuming q is piecewise constant over the time intervals,

$$\sum_{i=0}^{n-1} \frac{1}{|D\Phi_{1,t}|} q(1)\Delta t = I_1 - I^o(1),$$

which can be solved for

$$q(1) = \frac{I_1 - I^o(1)}{\sum_{i=0}^{n-1} \frac{1}{|D\Phi_{1,t}|}\Delta t}.$$

In the limit as $\Delta t \to 0$ the sum becomes an integral and we get the desired expression (which holds pointwise):

$$q(1) = \frac{I_1 - I^o(1)}{\int_0^1 \frac{1}{|D\Phi_{1,t}|}\ dt}$$

expressed in the coordinate system of image I_1. Here $\int_0^1 \frac{1}{|D\Phi_{1,t}|}\ dt$ corresponds to computing the map from each time point to the final time point.

Chapter 4

Model-based Guidance Computation

Guided (constrained) texture synthesis generates textures considering given guidance information, such as the model's regions and the direction and scale of the texture features in those regions. One problem with guided solid texture synthesis for anatomical structures is determining where to get the guidance. As discussed earlier, many textures on or within anatomical organs are oriented along or across the model, and they vary progressively depending on their position with respect to the model. Thus, the proposed approach in this dissertation is a method for computing the along-object (u), around-object (v), and through-object (τ) coordinates for any point in world space within each region inside the model. These coordinates should provide correspondence between different instances of the anatomical models. That is, corresponding points in different instances of an anatomic object, such as an anatomical structure in different patients, should have the same u,v,τ values.

In the literature, there are methods for computing the vector fields on and inside 3D models (Takayama et al. (2008a)). However, they need user interaction for the computation of these vector fields. Thus, they do not guarantee that these vector fields will be along or around the model. In addition, they do not provide consistency among different instances of the same model. To achieve that goal, I take advantage of m-reps. M-reps provide the three desired model-based coordinates (u,v,τ). They can also be used

to obtain consistent orientation fields and region classifications for different instances of the anatomic objects.

In this chapter the computation of the model-based orientation fields, scalar fields, material transition fields, and object layers are explained. Details of m-reps were presented in Section 2.4.3. In this dissertation it is assumed that an m-rep has been fit to the 3D object to be textured. Image segmentation methods that produce m-reps are described in Siddiqi & Pizer (2008). Merck (2009) describes the power of m-reps in volume visualization.

4.1 Overview

In this dissertation for computing the guidance information for a 3D model, such as orientation field and material transitions, the surface mesh of the model with its medial representation are used as input. First, using the fact that an m-rep implies the boundary, two model-based coordinates (u,v) are computed at each vertex on the surface mesh ($\tau = 1$ at all of these). These model-based coordinates are stored for each vertex in addition to the vertex's position (x,y,z). Secondly, the model-based coordinates at each voxel inside the model are computed using a standard scanning method, which interpolates the (u, v, τ) coordinates at each voxel. Finally, these model-based coordinates on the boundary and inside the volume along with the spokes from the medial sheet are used to compute different types of guidance information for that model:

1. Orientation field: Orientation vectors on and inside the 3D model surface are used to orient the structural features of textures on the surface and inside the model of the objects, respectively. In this dissertation, along, across, and through orientation vectors are computed using m-reps. The combination of these vectors can be used to obtain other kind of orientation fields, such as the oblique muscle layer of the stomach.

2. Material transition field: Material transition values are used to select the type of exemplar texture for synthesizing region-specific textures on the surface and inside the model of the objects. In this dissertation material transition values are specified using the along, across, and through coordinates for each primitive element in the model.

3. Scale field: Scale values are used to specify the size of the texture features in the exemplar textures for synthesizing textures inside the model. A change in size can be caused by many things. In this dissertation it is assumed that the narrow regions in the organs contain textures that have small scales and wide regions contain textures that have big scales.

These guidance fields are computed for both mesh vertices on the surface and the voxels inside the volume of the given 3D model. The orientation field of the surface is defined as a unit orientation vector for each mesh vertex, while the orientation field of the volume is defined as a unit orientation vector for each voxel.

4.2 Model-based Coordinate Generation

Assignments of a model-based coordinate for each vertex on the surface and for each voxel inside the volume are done using the following approaches:

1. **Model-based coordinates on vertices:** The model-based coordinates on the surface mesh can be computed using m-reps. Each boundary point carries a normal in the direction of the spoke at that position and two linearly independent directions in the tangent plane to the boundary corresponding to moving in the medial u and v coordinates respectively. For example, for the duodenum initially a set of surface points are generated from a grid of discrete medial atoms by following the spokes at each hub position to their surface tangencies. Each resulting

surface point is labeled with its source (u, v) as the (u, v) at the hub. Surfaces are recursively subdivided, and the u, v values at the new points are interpolated (if a new point is added between [3,7] and [3,8], its $[u, v]$ is [3,7.5]).

2. **Model-based coordinates on voxels:** Providing the (u, v) coordinates of each vertex on the boundary allows us to compute model-based coordinates inside the model. In order to compute a model-specific along, across/around, through map, which is called an X2U map in Merck (2009), a fast coordinate scan-conversion algorithm is used. First, repeatedly for many onion skins this method computes (u, v, τ) values at the vertices of the onion skins. Then, using clipping techniques it rasterizes the (u, v, τ) values into an (x, y, z) array. As a result (u,v,τ) coordinates at every voxel inside the model are obtained and stored as a lookup table. This method is proposed by Merck (2009). The details of this approach can be found in Merck (2009). The X2U map contains an object coordinate (u,v,τ) for each world-space (x, y, z) triplet inside the model. This coordinate system is precomputed and stored as a lookup table for each model. Figure 4.1 illustrates the X2U map of the scm both on the surface of the model and inside the model.

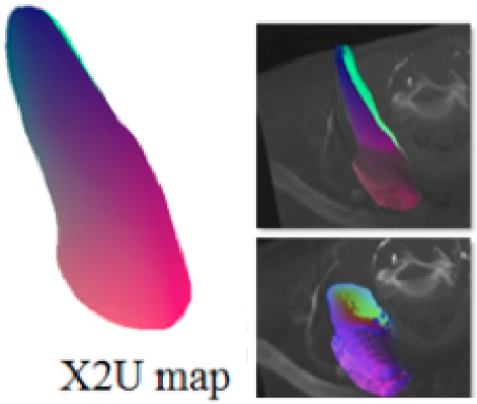

Figure 4.1: Direct display of the X2U map near the right sternocleidomastoid (scm) muscle. The red channel encodes u, the green channel encodes v, and the blue channel encodes τ. (Left) The X2U map is illustrated on the surface of the scm. (Right-top) The X2U map is illustrated on a layer through the model; (right-bottom) the X2U map is illustrated on a cross section across the model.

Beginning with a segmented model and the model-based coordinates, storing its X2U map has great advantages over other methods which encode only 3D distance information from the boundary, since this approach allows automatic computation of 3D model-based orientation information (du,dv,$d\tau$) and material transition information (called st) for illustrative purposes. In Merck (2009) X2U maps are used for integrating different sources of information into a single framework. In this dissertation they are used for computing the model-based guidance information.

4.3 Material Transition Field Computation

Material transitions are important for some of the anatomical models. There are basically two kind of transitions in the medical models of interest to my client. One is along the object, and the other is through the object. To obtain this transition for the models, a thresholding scheme is applied to the values in the X2U map.

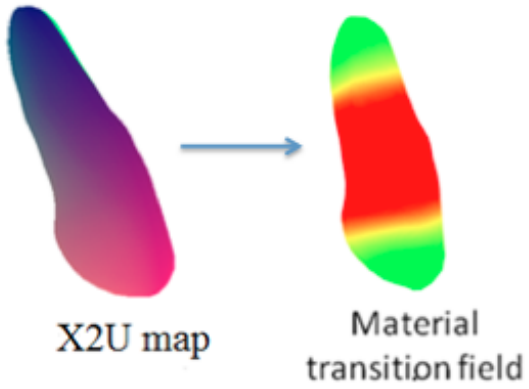

Figure 4.2: Along-object material transition field for the scm. Left: the X2U map on the model. The red channel encodes u, the green channel encodes v, and the blue channel encodes τ. Right: the material transition field along the model. The red channel encodes the first region, the green channel encodes the second region, and the yellow channel encodes the transition region between the first region and the second region.

1. Transition through the model: For the models in which the material is changing

according to the depth (e.g., bone), a thresholding scheme is applied on the depth information that is provided by the parameter τ. Using the fact that τ is zero at the medial sheet and 1 on the boundary, the regions are classified by the user by setting threshold values for τ (e.g., $0 < \tau < 0.5$ is the marrow region, $0.5 < \tau < 0.7$ is the interior bone region, and $0.7 < \tau < 1$ is the exterior bone region).

2. Transition along the model: For the models in which the material changes along the model (e.g., muscles), a thresholding scheme is applied based on user selection on the along-object coordinate that is provided by the parameter u (Figure 4.2).

After regions are classified using the model-based thresholding scheme, this information is used to compute a smooth transition field, st, for each voxel inside the model. The main advantage of this approach is that it gives consistent transition fields for different instances of the same anatomical organ.

4.3.1 Adding randomness to the transition regions

Although the thresholding scheme works well for obtaining uniform transitions along the desired directions, in Frank Netter, MD's illustrations, the transition is more random along that transition region. For example, in Figure 4.3 there is a randomness in transition field in addition to the distinction between the regions.

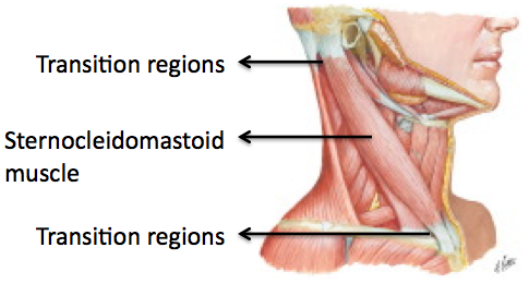

Figure 4.3: An illustration of the scm from Netter (2009). The transition between the red and white regions on the scm is not smooth. (@2011 Elsevier)

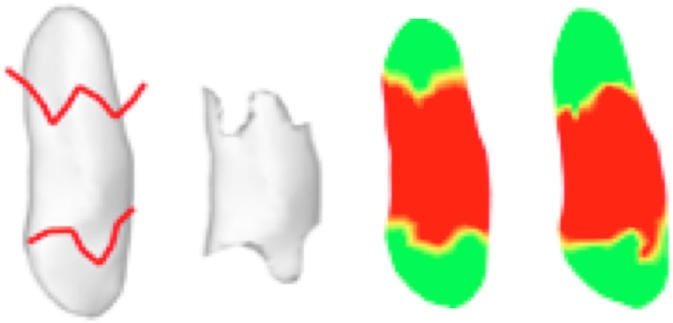

Figure 4.4: Along-object material fields can be modified by adding randomness around the thresholds via the sketch-based interface presented in Takayama et al. (2008a). (Left) two lines drawn on the model; (left-middle) part of the model between the lines; (right-middle) an example of regions specified for the model; (right) another example of regions specified for the same model.

To obtain randomness in the transition regions on the models, the sketch-based interface proposed in Takayama et al. (2008a) is adapted. In Takayama et al. (2008a)'s approach, the user is allowed to put colored points on the 3D model by clicking the mouse on the surfaces. The user has the ability to cut off the model, if s/he wants to place the points on the inner regions of the model. After assigning the points on and inside the model, the system computes a smooth material transition field inside the model. In this dissertation the position and colors of these points are obtained from medial-coordinates and the lines drawn by the user. The steps of the proposed approach in this dissertation are as follows:

1. Load the model with its medial representation: This step consists of loading both the 3D mesh of the model and the model-based coordinates of each vertex of that model.

2. Specify model-based regions along and through the object: This step is done using ranges of u. (The method can be extended to ranges in τ as well.)

3. Assign a region label value to each specified region of the object surface: These label values will be used in step 6 as constraints to compute a smoothly varying color field across the voxels within the object. Here a fixed label value is assigned

to all vertices whose u value falls in the range specified for that region.

4. Add randomness to the boundaries between adjacent regions: This step is achieved by drawing a piecewise linear curve around the thresholded region boundaries using the sketch-based interface. These curves are applied at all depths relative to the screen (Takayama et al. (2008a)). The subdivision of the mesh into subregions is updated based on the position of the curves.

5. Update the label values of the mesh vertices if their regions are changed: If the region of a vertex is changed from region 1 to region 2, then change the label of that vertex from 1 to 2.

6. Use interpolation to find a real label value for each voxel inside the model; these will be used to interpolate the textures between the regions . Here, thin plate spline interpolation (Turk & O'Brien (1999)) from the object surface vertex coordinates is used to find a smoothly varying scalar field in the 3D space.

Using this approach, different region boundaries can be obtained near the same thresholded region boundaries. Figure 4.4 illustrates some transition fields obtained using this technique.

4.4 Scalar Field Computation

For some of the objects, the scale of the textons inside the textures can change depending on the region. This change can be due to the thickness in depth of the region or due to hypertrophy, which is an increase in cell size. In this dissertation the first case is considered.

The thickness of each region is computed as guidance information using the medial representation. To achieve that, the length of the spoke is used for encoding the thickness of a region inside the model. This is done by computing a X2L (X to length) map which

stores for each voxel the length of the spoke passing through it. For instance, for a voxel that has medial coordinates $(u, v, 0.5)$, the value that is going to be stored for that voxel is the distance between the voxel at $(u, v, 0)$ and the voxel at $(u, v, 1)$. Computation of X2L is done by the same kind of rasterization as with X2U. Figure 4.5 shows the X2L map for a pyramid model.

Figure 4.5: Scale field (X2L map) for a pyramid model. This model gets narrower when you move from bottom to top. Intensity values correspond to the narrowness of the regions.

4.5 Vector Field Computation

The MGTS framework consists of oriented surface and solid texture synthesis methods. Thus, orientation vectors of each vertex and voxel have to be computed to be used as input for these methods:

4.5.1 Surface vector field

For anatomical objects the model-based orientation field on the surface mesh is computed using medial coordinates (u, v, τ) at each vertex (x, y, z). To achieve that, directional derivatives of $[u, v]$ are then taken with respect to $[x, y, z]$. Assuming that the underlying samples are organized so that u runs along the object and v runs across it (typically enforced by construction), the direction du ends up representing the local surface direction

along the object and the direction dv ends up representing the local surface direction across the object (except at the boundary points where u or v is 0 or 1). When a shape is loaded with its medial representation, first du and dv are computed using the gradient with respect to u and v at each vertex, respectively. The dv vectors are reversed on one side of the model, since the orientation of dv vectors should traverse around the model. Then a basis for the tangent plane at each vertex is computed by projecting the local du and dv directions. These projected vectors, du_{proj} and dv_{proj}, may not be orthogonal to each other. Thus, du_{proj} and dv_{proj} are orthogonalized to give $du_{orthogonal}$ and $dv_{orthogonal}$. This orthogonal basis can then be used as the so-called "tangent" and "bitangent" directions for synthesizing the textures on the surface.

Fig. 4.6 shows du and dv for two different instances of duodenum model.

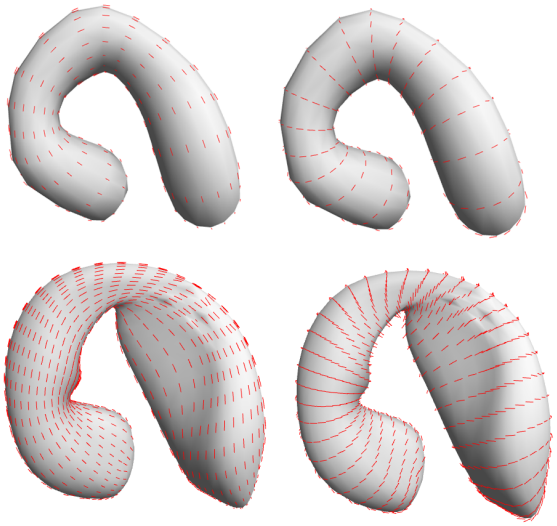

Figure 4.6: Vector fields for two different instances of the duodenum. (Left) Along-object vector field; (right) across-object vector field

4.5.2 Volumetric vector field

For anatomical objects the model-based orientation field inside the model is computed using medial coordinates (u, v, τ) at each vertex (x, y, z) of many layers through the model. This is achieved using the following steps:

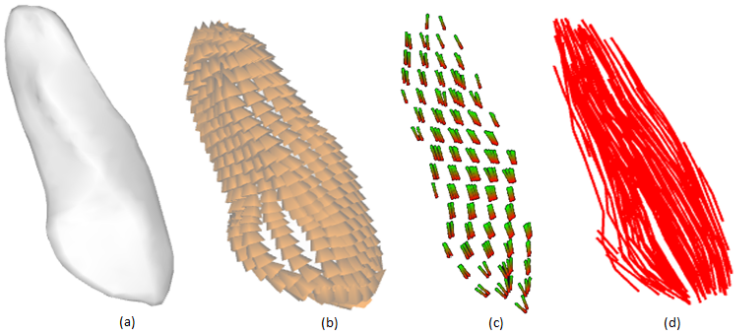

Figure 4.7: Along-object volumetric vector fields computed for the scm. (a) The scm surface model; (b) along-object vector field on the outmost layer; (c) along-object volumetric vector field; (d) streamlines computed from the vector field

1. Computation of layers: For a user chosen n, n surface layers are computed through the model using linear interpolation according to the τ values. (u, v) values for each vertex on these layers are also stored with each layer.

2. Computation of the vector fields on each layer: Vector fields (along and across the object) are computed for each surface layer. This is achieved automatically from the gradient of the (u, v) coordinates at each vertex of these layers.

3. Interpolation of vector fields: Orientation vectors on the nearby mesh vertices are blended to set the orientation vector of boundary voxels. To achieve that, the interpolation method proposed in Takayama et al. (2008a) is adapted.

4. Projection of the layer vector fields to the tangent planes: It is assumed that the directions should be parallel to the model's surface layers. If the vectors at each voxel are not tangent to the surface of the model, the vectors associated to the mesh vertices are projected to their tangent planes.

5. Smoothing of vector fields: A Laplacian smoothing is applied to the voxels inside the model in order to obtain the final volumetric vector field. This is achieved using the method presented in Takayama et al. (2008a).

6. Normalization of vectors: Vectors at each voxel are normalized since magnitudes

of the vectors are not considered in guided solid texture synthesis.

Figure 4.7 shows the along-object vectors for the scm.

It is hard to compute the $d\tau$ orientation field using this approach, since you have to specify many surfaces and the orientation fields on these surfaces along the object. Instead, $d\tau$ is computed by finding the orientation field that is orthogonal to the du and dv orientation fields.

Chapter 5

Model-based Surface Texture Synthesis

Surface textures are important for illustrating some anatomical organs, especially those which have simple, smooth shapes made from a simply curving cross section gradually varying along a smoothly curving axis. Such objects include bones (not at the end of the arm/ leg/ finger bones), muscles, tubular structures (e.g., the stomach and the intestines), and many others.

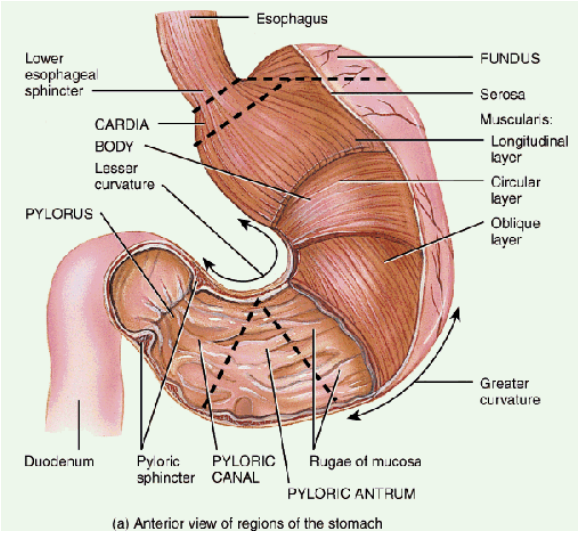

Figure 5.1: Different layers of the stomach (http://www.rivm.nl)

For some applications, like virtual endoscopy and virtual colonoscopy, rendering of the single outside organ surface could be enough for identification purposes. However, many organs have multiple tissue layers of interest near their surfaces, and these layers

are mostly illustrated separately with different textures in textbook illustrations. Each of these layers can be composed either of the same material with different orientation and scale, or of different materials with different tissues. For the digestive system, the common layers in the organs are the mucosa layers, the muscular layers, and the serosa layers. There is usually more than one muscular layer in these organs, and each layer has differently oriented muscle fibers. For instance, the stomach has three layers of muscle: innermost oblique, middle circular, and outer longitudinal. Figure 5.1 shows an illustration of the layers of the stomach.

To obtain illustrative visualization of these organs, surface textures for each layer should be generated, and a means for exposing these inner layers to view must be provided. Given 2D exemplar textures and 3D surfaces, there exist several approaches for synthesizing anisotropic textures on the surfaces. Commonly these methods (Praun et al. (2000)) need user intervention to create the orientation field and layers on the surface, and for that reason they cannot provide orientation correspondence between different instances of the same model. To solve these problems, this dissertation proposes an automatic method called model-based texture synthesis, which generates consistent anisotropic surface textures for patient-specific anatomical models using illustrative model-specific exemplars. This method relies on m-reps to generate oriented textures on the surface of the object. The main advantage of using m-reps here is that it gives model-based orientation fields on the surface of the models for each patient data without any user intervention. These vectors are consistent for different instances of the model. M-reps also provide the layers along the depth of the model. These layers are important for the illustration of different tissue layers for that specific model. Using the orientation field on each layer of the model, the model-based surface texture synthesis generates surface textures for that layer.

Given a 3D model M with its medial representation and 2D exemplar textures, the textures on different layers of the model M are synthesized using an exemplar-

based texture synthesis technique. Texture synthesis on a 3D surface layer of a given anatomical model is achieved by combining and extending the mesh partitioning and parameterization approaches proposed in Gorla et al. (2003) with the optimization-based texture synthesis algorithm summarized in Section 5.3. In particular, the framework has two main components: (1) model-specific guidance and subdivision information computation and (2) texture synthesis.

1. Model-specific guidance and subdivision information computation: The aim of this component is to preprocess the object model. These steps are completely independent of texture synthesis to make texture synthesis faster. The preprocessing consists of a sequence of steps. These steps need to be applied only once per model to support any number of texture synthesis. In addition, this precomputation provides the ability to apply different pixel-based texture synthesis approaches without modifying the mesh related parts.

   (a) Computing the vector field: This step consists of computing orientation fields for different layers of a specific anatomical model. The details of how this is achieved were presented in Section 4.5.1. There are two particular types of orientation fields that can be generated using m-reps:

      i. Along-, across-, through- object vector fields over the surface: They are obtained using the model-based coordinate system, and they provide a common basis for any oriented texture.

      ii. Combination of along-, across-, and through- object directions: They provide a specific local orientation frame. For instance, the orientation field for the innermost oblique muscle layer in the stomach can be obtained by combining the around- (circular), and along- (longitudinal) object orientation fields.

   (b) Partitioning the 3D surface mesh: In this step contiguous patches of the

polygons of the model are computed. The partitions should be as nearly planar as possible in order to prevent distortion of the texture when it is mapped on the 3D surface. In addition, the partitions should have similar sizes to simplify the texture mapping.

(c) Projection onto the 2D reference plane, the computation of which is described in Section 5.1: In this step the triangles and vector fields in each patch are projected onto a reference plane of that patch in order to do texture synthesis in a 2D plane.

2. Texture synthesis: The aim of this component is to create textured surface layers of a patient-specific model using mesh-related inputs from the first component, 2D exemplar textures, and guidance information.

   (a) Synthesizing texture over each of the projected 2D patches: In order to maintain pattern continuity across the patch boundaries (seams), a pixel-based approach is used for texture synthesis. As described in Section 5.3, the approach presented in Kopf et al. (2007) is extended for synthesizing textures with orientation fields. In the results presented in this section the anisotropic textures are applied such that their dominant orientation is everywhere aligned with the direction field specified using the model-based coordinate system.

   (b) Mapping synthesized textures onto the 3D surface: The synthesized texture for each patch is mapped on the 3D surface mesh. Alpha blending is done around the border of each patch in order to achieve continuity across the seams.

The layers of the textured models are visualized using the model-based clipping technique available in the MGRView framework (Merck (2009)). In that approach first each surface layer is textured with material-specific textures. Then, a range of u and v values are used to define a clipping region on a selected layer. Finally, the vertices that

have the (u, v) values in the specified range are deleted from that layer. Figures 5.9 and 5.10 illustrate the results obtained using this approach.

5.1 Surface Partitioning

In this stage the 3D surface mesh is partitioned into a minimum number of planar patches which have similar sizes. It consists of two main steps (More details can be found in (Gorla et al. (2003))):

- Initially, the 3D surface is partitioned using a greedy algorithm. In this algorithm a randomly-selected triangle (reference plane) is assigned to its own group, and then new triangles are added to this group based on the angle and distance between the new triangle and the other triangles in the group. Thresholds are used for both angle and distance for this approach.

- An optimization step is performed to reduce the number of patches and the number of triangles that are oriented at a sharp angle to the plane onto which they will be projected.

5.2 Projection of Surface Patches

The triangles in each patch are projected onto their common reference plane. Using the coordinates of the projected vertices in the reference plane coordinate system, the texture coordinates are computed. In addition, the neighboring patches are projected onto the reference plane since the textures synthesized for the patches in the neighborhoods are used as boundary conditions in texture synthesis. Figure 5.2 illustrates the process.

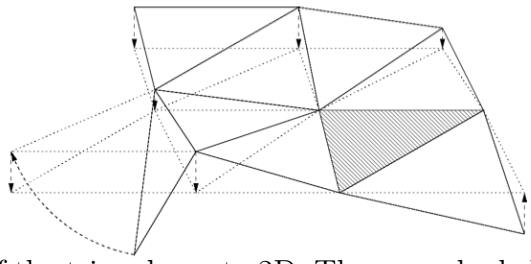

Figure 5.2: Projection of the triangles onto 2D. The grey-shaded triangle is the reference triangle. Its normal defines the plane onto which the other triangles in the patch will be projected. This figure is taken from Gorla et al. (2003).

5.3 Texture Synthesis

Textures are synthesized for each patch of the surface layer. In order to synthesize texture for a patch, first a 2D image $\mathbf{x}^0$ is created for that patch. Then initial values are assigned for each pixel in $\mathbf{x}^0$ from two different sources: (a) intensity values of the pixels in the projection of the triangles inside that patch; and (b) intensity values of the pixels in the projection of the triangles in the neighborhood patches.

For anisotropic textures, per-pixel texture orientations $\mathbf{v}$ are computed by first projecting the vector values at the vertices of the triangles onto the reference plane and then interpolating the vector values for each pixel inside the triangles using barycentric coordinates.

Texture synthesis is done by adapting the approach presented in (Kwatra et al. (2005)). This approach is summarized in Section 2.3.1.3. Given the 2D image $\mathbf{x}^0$ and the projected vector field $\mathbf{v}$, textures are synthesized for the image $\mathbf{x}^0$ by considering the vector field $\mathbf{v}$. During texture synthesis, textures synthesized for the boundary patches are taken into consideration in order to maintain the continuity of the texture pattern across seams at the patch boundaries. For each patch P_i on the surface the following steps are applied in order to achieve that goal:

1. Create a 2D planar texture T_i for patch P_i.

2. Project the triangles that are in P_i onto that planar texture and assign their colors

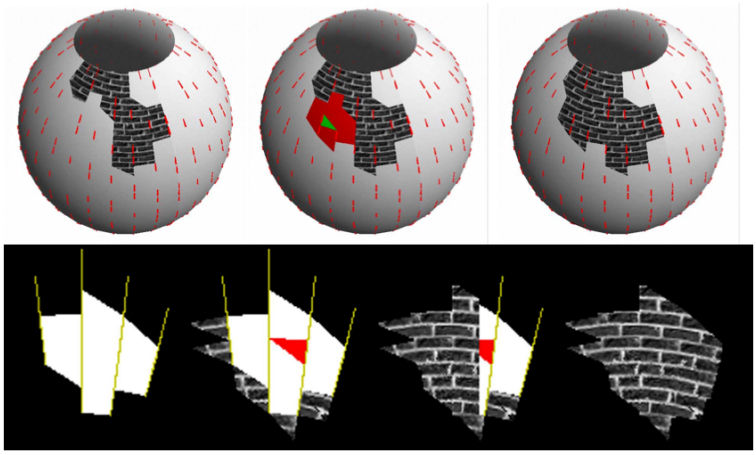

Figure 5.3: Steps of texture synthesis for each patch. The figure is taken from Gorla et al. (2003). Upper row: Texture synthesis for a selected patch P_i on the surface. (Left) synthesized textures on the first two patches; (center) the triangles on the selected patch (the patch is shown with red, except for the green triangle that is the reference plane of the patch); (right) textured surface after the texture is synthesized for the third patch. Lower row: texture synthesis on the 2D projected plane for the selected patch. (Left) the planar projection of the selected patch; (left center) the selected patch with the boundary conditions provided by the neighboring textured triangles rotated into the plane of the patch; (right center) midway through the texture synthesis process (synthesis is proceeding from left to right); (right) the complete synthesized patch.

to white.

3. For each neighbor patch P_j of P_i, check whether a texture is synthesized for that patch before. If so, project this texture onto the plane of texture T_i.

4. Synthesize texture T_i.

The lower row in Figure 5.3 illustrates these steps.

For anisotropic textures the synthesized texture is locally constrained into alignment with the vector field $\mathbf{v}$ over the surface. This alignment is achieved by rotating the neighborhoods at each pixel in the image using the orientation vector $\mathbf{v}$ at that pixel. This rotation of neighborhood is done both in the search phase of the approach (in the M step) for finding best-matching neighborhoods and in the optimization phase of the approach for calculating the intensity value at that location. Figure 5.3, obtained from Gorla et al. (2003), illustrates the texture synthesis process for a selected patch. After

the texture is synthesized for the selected patch, the 2D image is stored as a result for that patch. This image is mapped onto the 3D surface for rendering the texture.

5.4 Results

Given models, which can be generated automatically using methods from image analysis, the presented framework is fully automatic and does not require any user interaction during either the model-based part or the texture synthesis part. The user only specifies the regions in u and in τ. There are several parameters in the system that can be set by the user, such as the size of a single patch and the size of the neighborhoods.

In this section the results of the framework on three different organ models from different patients are presented. These models are obtained from real patients' segmented data. The exemplar textures are created using imaging tools by considering Frank Netter's illustrations. Below are the selected anatomical organ models:

- **Duodenum:** This model consists of many layers: longitudinal muscle, circular muscle, submucosa, mucosa, and circular folds. Among these layers the first two muscle layers and mucosa layer are oriented along or across the duodenum.

- **Sternocleidomastoid muscles (scm):** These muscles attach to your top two ribs and to several neck vertebrae, resulting in muscle fibers of varying lengths. The muscle fibers are elongated along the object.

- **Stomach:** This is a J-shaped hollow muscular organ. It has mucosa, submucosa, and muscularis externa layers. The muscularis externa has its own three layers, which are the inner oblique layer, middle circular layer, and outer longitudinal layer.

Figure 5.4 shows the results of the approach for the duodenum model. Textures for the longitudinal muscle layer, circular muscle layer, and mucosa layer of the duodenum

are synthesized for that duodenum model. The result for each muscle layer illustrates the arrangement of the muscle fibers using the exemplar texture, the model-based orientation field, and the model.

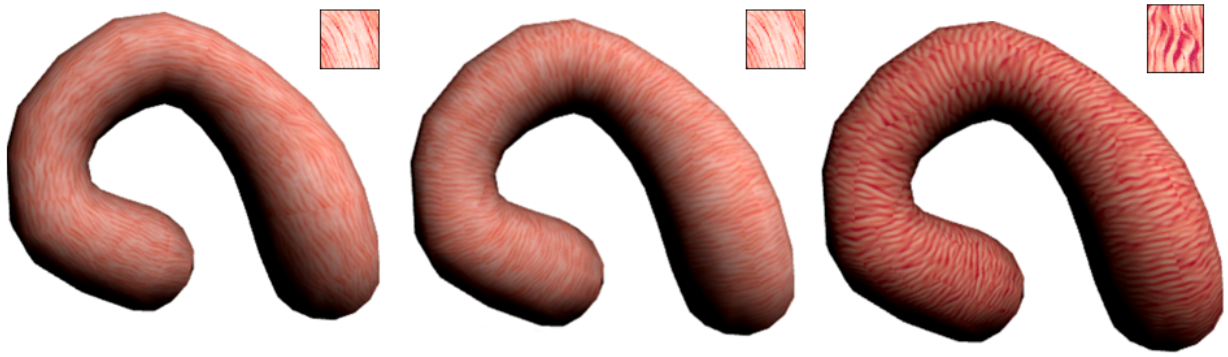

Figure 5.4: Three different layers of the duodenum: (left) longitudinal muscle layer, (middle) circular muscle layer, and (right) mucosa layer. Before using the input texture as an exemplar, its orientation has been adjusted.

Figure 5.5 shows the results of the approach for the submucosa layer of the duodenum model. This layer is illustrated with isotropic textures as opposed to the other layers of the duodenum, which are anisotropic.

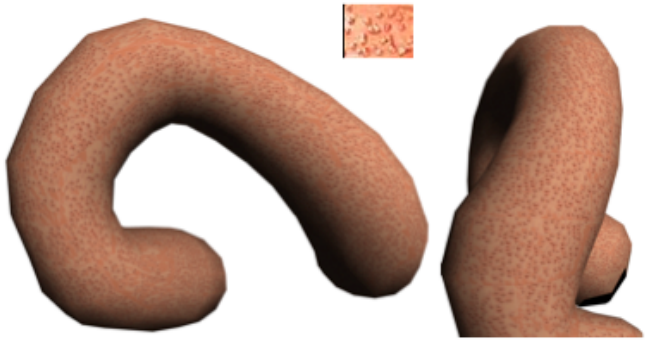

Figure 5.5: Submucosa layer of the duodenum. As opposed to the other three layers, this layer is isotropic.

The synthesis approach is applied to generate texture on different models using the same exemplar texture. In Figure 5.6, the models, stomach and scm, are textured using the model-based orientations along the objects.

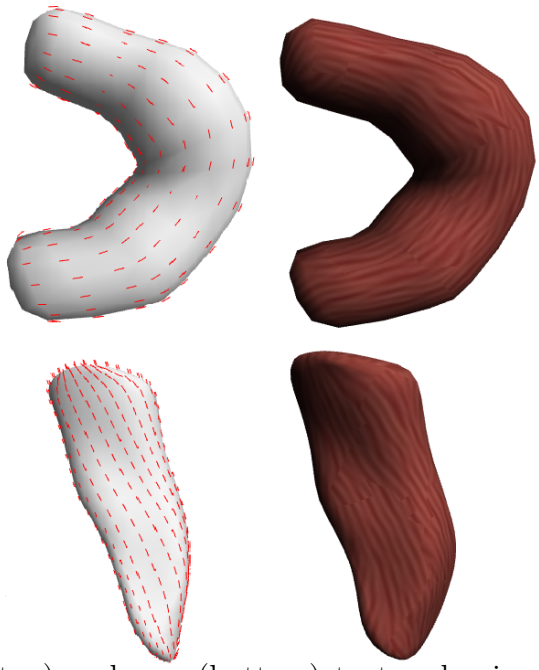

Figure 5.6: Stomach (top) and scm (bottom) textured using along orientation fields. (Left) Orientation fields; (right) textured results.

Using different combinations of along- and across-model orientation fields, other orientation fields can be obtained on the surface of the models. For example, for the oblique muscle layer of the stomach, the sum of the along- and across- orientation fields are used to obtain the oblique orientation field for that layer (Figure 5.7).

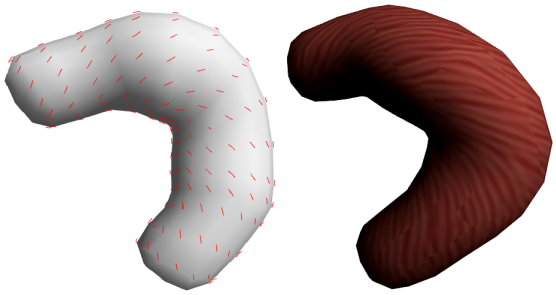

Figure 5.7: Oblique layer of muscularis externa on stomach model. (Left) Orientation field; (right) textured result.

In order to show the orientation correspondence, textures for different instances of the same model are synthesized using the along-object and across-object orientation fields. The orientation fields for that model are shown in the previous section in Figure 4.6. As can be seen in Figure 5.8, the presented framework generated consistent textures

on the models without any user intervention for orientation field and texture generation.

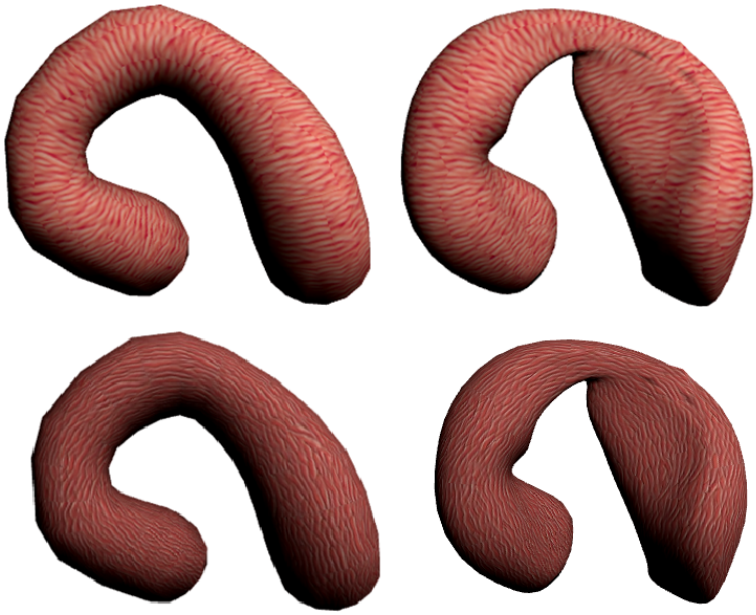

Figure 5.8: Two instances of the duodenum textured with along and across orientation fields.

To obtain better understanding of complex 3D shapes and interrelations between 3D objects, the textured 3D anatomical models have been integrated inside real patient 3D data display (Merck (2009)). In Figure 5.9 and Figure 5.10 the duodenum inside the patient data is shown in MGRView framework presented by Merck (2009). Three different layers of the duodenum (longitudinal, circular muscle, and internal rugae) are combined to obtain the illustration of that model. Using this approach, patient-based anatomic illustrations are obtained with model-based oriented textures and segmented 3D surfaces.

In addition to synthesizing oriented textures, the presented surface texture synthesis approach can also be applied for synthesizing isotropic textures, such as the surface of the bone shown in Figure 5.11.

These results have demonstrated the main advantage of the presented framework, that it can perform texture synthesis on different patients' organs without any user

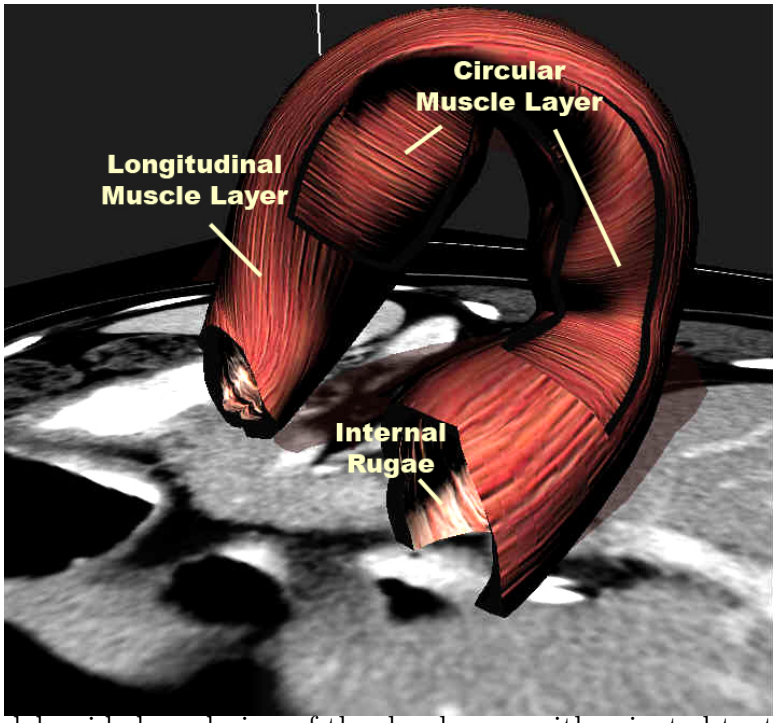

Figure 5.9: Model-guided rendering of the duodenum with oriented textures for context cues. The duodenum is visualized inside the real 3D patient data using MGRView (Merck (2009)).

intervention for specification and computation of the model-based vector field. In addition, the results show that the method successfully provides orientation correspondence between different instances of the same model, which is a requirement for producing consistent results in anisotropic texture synthesis. This framework can be used for visualizing different anatomical structures that consist of many layers. In addition, in the future this framework can be applied to variable scenarios in order to reflect differences in individual patients.

Figure 5.10: Textured duodenum visualized in MGRView (Merck (2009)) with respect to two clipping planes.

Figure 5.11: Surface of the bone is textured.

Chapter 6

Model-based Solid Texture Synthesis

Many anatomical objects have slowly changing variations in local material properties, such as material type, scale, and orientation, over the surface of the model and inside the model. In many of the hand-drawn medical illustrations commonly used for anatomy reference, these features are illustrated mostly with textures. The artists used textures that suggest features arranged along, across/around the object, and through the object. For example, the bone has hard exterior, soft interior, and soft marrow interior regions, and these regions are illustrated with different material textures. In addition, artists also illustrate the inner regions of models by putting textures on the clipped sections of the selected regions.

In order to provide this behavior for the illustrative visualization of patient data sets in a 3D environment, in this dissertation a method is presented for synthesizing patient-specific progressively variant solid textures for segmented patient data. The proposed approach is based on the non-parametric exemplar-based techniques used in Kopf et al. (2007), in which uniform cubic solid textures are synthesized. The major contribution of the proposed method is providing an extension to this method to efficiently support object-specific progressively changing anisotropic textures. This part of my dissertation is published in Kabul et al. (2010).

My method produces solid textures that are constrained by a 3D vector field as well

as by material information specific to the patient's anatomical structures. To capture model-based directions, material transitions, and depth of layer information consistently inside the model for the segmented patient data, the texture synthesis approach utilizes the model-based coordinate system presented in Section 2.4.3. Given a surface model with its medial representation and 2D exemplar textures, the proposed method first computes the model-based vector fields and region masks, and then it uses them to synthesize model-based progressively changing solid textures inside the models. As opposed to the approach presented in Takayama et al. (2008b), control of texture variation and texture orientation is accomplished by generating the textures "in-place" for a model in world space instead of in model space. This approach prevents the possible seams that might be caused by bending and twisting a standard texture cube according to a particular model.

My method considers orientation and material information for each voxel as the means of constraining texture synthesis. The 2D input exemplar textures suitable for synthesizing anatomic structures are taken from illustrated sources such as Netter (2009) or from specifically designed anatomic texture catalogs. As indicated in the previous chapters, the appearance of the texture undergoes changes depending on the model. To solve this problem, model-based texture synthesis control is added to restrict the appearance of the textures for clipping planes that are rotated based on the orientation of medial axis.

Some of the structural textures consist of structural primitives (textons), such as glands in the prostate. The number of textons and the size of the gap between them can change depending on the organ and depending on the patient. In this framework the user has the ability to control these features in the synthesized texture via a new texton matching technique. In addition, the synthesis of structural solid textures from 2D exemplar textures needs to accommodate control of the shape of the structures in 3D. To achieve that, a texton-based solid texture synthesis is proposed for controlling

the construction of 3D textons via a medial representation of textons in 2D exemplar textures.

The results of the approach are demonstrated on several examples including the thyroid, the sternocleidomastoid muscle (scm), the prostate, and the mandible. The method has been shown to work for the anatomical organs in the head and neck region. However, the method is extensible to different instances of many organs, such as muscles. The goal of this approach is eventually to be able to fill a space containing many organs with synthesized solid textures to create an illustrated look for a particular patient. In this dissertation, all of the organs could not be rendered with textures due to the lack of segmented models.

6.1 Overview

The model-guided texture synthesis (MGTS) method synthesizes progressively changing solid textures specific to a model of an anatomical object by considering the orientation and variation of the textures inside the model. The purpose of this method is to obtain illustrative renderings of patient-specific anatomical structures. The input to the system consists of a 3D triangular surface mesh, a model-based coordinate system for the model (in our method obtained from a medial model of the object), and a predetermined set of 2D exemplar textures. The output is a solid textured model. Thus, the model-guided texture synthesis is a post-segmentation process. Our method also consists of synthesizing solid textures according to the guidance information, such as the vector field and the depth of layer.

As with the 2D texture synthesis described in Chapter 5, in this method the 3D guidance vector field is obtained from a medial representation called 'm-reps'. M-reps nicely provide a 3D along-object, across-object, and through-object parameterization on and within the model. Details of this approach have been explained in Section 4.5.2.

The two features considered in the texture synthesis step are the guidance vector field and the variation of the textures inside the model. Depending on the anatomical structure, the texture may vary along the depth of the model or along (or around) the model, whether on the surface or its interior. This variation information is obtained from the user-annotated model-based coordinate system from the first stage and is used to guide the solid texture synthesis.

For some of the anatomical models, the textures look different depending on the orientation of the clipping plane. For example, the texture on the outside of the sternocleidomastoid muscles (scm) looks like a stream of lines, whereas when you cut the scm, you see a circular blobs texture. In order to obtain this effect in our visualization, the 3D model-based guidance information computed in stage 1 is again used.

In the following sections I will discuss the solid texture synthesis methods proposed for generating oriented, progressively-changing, and texton-constrained solid textures.

Notation: In the following sections e denotes the input exemplar texture, ef is the feature image (e.g., signed distance field) of e, em represents the medial coordinates of each pixel in e, and et stores transition of the material information (see Section 6.2 for explanations of ef and et). Also, s refers to the synthesized solid texture. Three model-based coordinates at each position in the object are defined as along-object (u), around-object (v), and through-object (τ) coordinates. Three special orientations at these positions are $du = \nabla u$, $dv = \nabla v$, $d\tau = \nabla \tau$. These three are not necessarily orthogonal in x, y, z space. The neighborhood of the voxel w in the slice perpendicular to the i^{th} spatial axis ($i = x$, y, or z) is represented by $s_{w,i}$. The neighborhoods for the exemplar texture and for the synthesized solid texture at voxel w are denoted by e_w and s_w, respectively. For region-based textures, st denotes the 3D transition field inside the model, which is computed using the model parameterization.

6.2 Texture Features For Solid Texture Synthesis

As explained in Section 2.1, a texture can be represented using different techniques depending on the type of the texture. Texture features are important in texture synthesis for representing the exemplar textures and generating the output textures. Most of the texture synthesis approaches use statistical (intensity-based) and region-based (structural) features of the textures. Intensity-based features model the joint statistical properties of features extracted at pixel locations; thereby they are useful to preserve the color information in the synthesized texture. Structural features model texture as a layout of regions; thereby they are important in synthesizing the structural layout of the output texture. Many approaches use different features such as edges or signed distance fields to these edges as structural texture features. However, these features do not provide a good parameterization of the structural primitives in the texture. For instance, a signed distance field gives the through-region parameterization of the primitives. It does not provide the along-region parameterization, which is important information when you synthesize solid textures.

In this dissertation the textons, which are the structural primitives, in the textures are themselves represented using an approximate medial representation. This parameterization provides the along and through coordinates of each pixel inside each texton in the exemplar textures. These texton parameters are used to restrict the search phase of the synthesis method. In this section, the computation of textons, their parameterization, and their histograms are presented. The details of the synthesis process are explained later in Section 6.3.

6.2.1 Textons

Color textons have been widely used in texture recognition problems. Although there exist many methods for finding the textons automatically in a given texture (Ahuja &

Todorovic (2007)), these methods do not work for all textures. The research in this area is beyond the scope of this dissertation. In MGTS textons are specified using a thresholding scheme and manual manipulation of the thresholded image. After that a method to label the connected components in the binary image is applied. Using this approach, a texton segmentation is obtained for each given exemplar texture.

6.2.2 Signed distance field

Signed distance is important for the computation of the inner coordinate system of the textons in the exemplar texture. Given the thresholded exemplar texture, first an edge detection technique is applied to obtain the edges of the texture. Then the signed distance field is computed on that image (Figure 6.1).

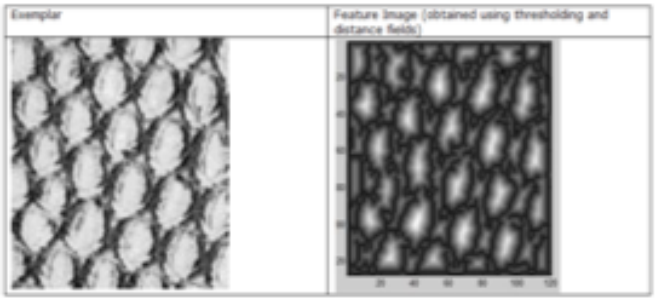

Figure 6.1: (Left) Texture image; (right) signed distance field of the texture.

6.2.3 Medial representation of textons

In order to find the approximate medial coordinates of each pixel in each labeled texton of a given texture, I utilized the following steps:

1. Threshold the given image to obtain the binary image that contains the textons (Figure 6.2-(b)).

2. Label each texton of the given binary image (Figure 6.2-(c)).

3. Compute the signed distance field of the given binary image. Store it as the τ coordinates image (Figure 6.2-(d)).

4. Calculate the medial skeleton of each texton.

5. Remove the branches in order to obtain a single medial axis.

6. Parameterize the medial axis using along-object coordinates u.

7. Classify the pixels in each texton of the input image based on their distances to the medial axis. Assign the u value of the closest point on the medial axis to that pixel. Store the u values to form the u coordinates image (Figure 6.2-(e)).

8. Classify the pixels in each texton of the input image based on their u values. Assign the distance value of the closest point on medial axis to that pixel. Store the distance values to form the $r\tau$ coordinates image (Figure 6.2-(f)).

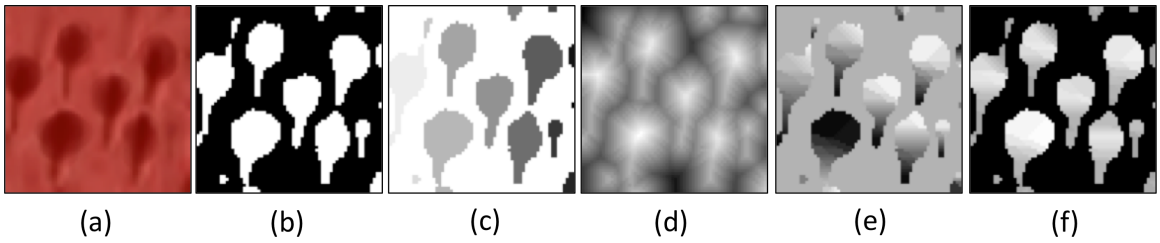

Figure 6.2: Medial representation of a 2D texture. (a) A 2D texture; (b) binary thresholded image of the given texture; (c) image that contains the labeled textons; (d) signed distance field image computed using binary thresholded image; (e) along-texton parameterization for each texton in the given image; (f) thickness of each pixel based on the along-texton coordinate.

6.2.4 Histograms

A histogram is one way to represent the distribution of data in a texture in a compact way. Histograms on certain values can be useful to represent the properties that are important for the target application. For instance, a histogram of texture intensity

values is an important property for preserving the global statistics of the 2D example texture in the synthesized texture. More details of the histogram types used in texture synthesis are presented in Section 6.3.3.1.

6.3 Optimization-based Solid Texture Synthesis

The intensity value at a pixel/voxel in a 2D/3D texture is highly dependent on the intensity values of neighboring pixels/voxels around that pixel/voxel unless the texture is a random noise. This dependence can be explained using a Markov Random Field (MRF) (Cross (1980)). In such a model the probability of a pixel given the rest of image is equal to the probability of that pixel given the neighborhoods of that pixel (P(pixel p|rest of image) = P(pixel p|neighborhood of p)). An important requirement for both 2D and solid texture synthesis is to satisfy this property in the synthesized texture. This is typically achieved by searching the input exemplar texture for all sufficiently similar neighborhoods and selecting one match at random.

The MGTS framework uses 2D exemplar textures as input (since it is hard to find 3D exemplar textures). Thus, the neighborhoods, which are used in the search phase, are constructed in 2D in order to compare the neighborhoods in the synthesized texture and in the exemplar texture. The number of neighborhoods and their orientations is an important parameter in solid texture synthesis, since the best-match search is done for the values in these neighborhoods. Although increasing the number of neighborhoods can give better results, the computation time for the neighborhood search will increase in that case. Thus, the orientations that are going to be matched to the given exemplar/s should be selected. This selection can be done depending on the type of the output texture. If the solid texture is isotropic, it should be similar to the exemplars on arbitrary slices through the entire volume. Thus, using three orthogonal directions can be sufficient. If the output texture is anisotropic, these directions of the neighbor-

hoods should be chosen such that the exemplar for the selected direction represents the intended appearance in that direction.

Texture synthesis consists of two main steps: a search phase and an optimization phase. The approach presented in Kopf et al. (2007) for synthesizing solid textures is also a two-phase optimization method that tries to minimize the sum of differences under the L_p norm between each local neighborhood $s_{w,i}$ $(i = x, y, z)$ of a voxel w in the output texture and a corresponding neighborhood $e_{w,i}$ $(i = x, y, z)$ in the 2D example texture e. Initially the colors are randomly generated for each voxel inside the cube from the 2D exemplar textures. Then in the optimization stage the matching neighborhood of each voxel is determined by optimizing the energy function E_t:

$$E_t(s; \{e\}) \approx \sum_{w} \sum_{i \in x,y,z} ||s_{w,i} - e_{w,i}||^r = \sum_{w} \sum_{i \in x,y,z} weight_{w,i,u} ||s_{w,i} - e_{w,i}||^2 \qquad (6.1)$$

In the search stage the best matching neighborhood from the exemplar texture is found for each voxel. The details of this approach can be found in Kopf et al. (2007).

The main problem in Kopf et al. (2007)'s method is the blurring problem in the synthesized texture. This is mainly caused due to the averaging done for the computation of the voxel values in the output texture. To solve this problem, another approach based on Kopf et al. (2007) is proposed in Chen & Wang (2010). In this approach the optimization framework is achieved using the k-coherence search presented in (Tong et al. (2002)) and the discrete solver presented in Han et al. (2006). To avoid blurring, this approach requires that all the voxel colors be copied directly from the exemplars. This approach also introduced two new histogram matching methods (for details see Section 6.3.3.1) in the re-weighting scheme to ensure that all the voxel colors are copied uniformly from the exemplars. This approach produces isotropic cubic textures. Figure 6.3 shows the results of this approach for the thyroid. In this dissertation the method proposed in Chen & Wang (2010) is extended for synthesizing anisotropic, region-specific

solid textures.

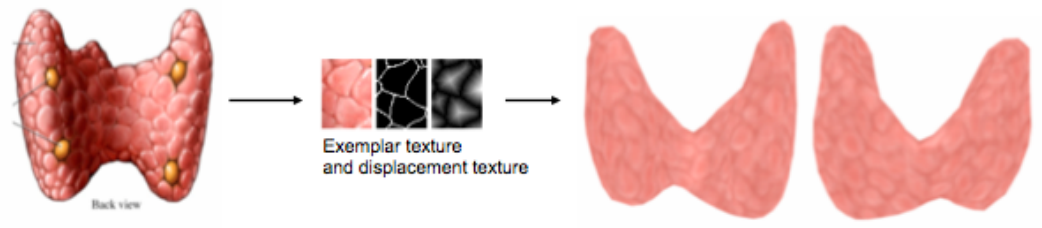

Figure 6.3: Isotropic texture is synthesized for the thyroid using a sample exemplar texture from an illustration.

6.3.1 Optimization phase

In this phase of the approach to minimize the energy, the discrete solver proposed in Chen & Wang (2010) is used instead of the least-squares solver proposed in Kopf et al. (2007). In a least squares solver, the updated voxel is computed using

$$s_w = \frac{\sum_{i \in x,y,z} \sum_{u \in N_i(w)} weight_{u,i,w} e_{u,i,w}}{\sum_{i \in x,y,z} \sum_{u \in N_i(w)} weight_{u,i,w}} \tag{6.2}$$

In contrast, in the discrete solver, s_w is calculated first, and then $e_{u,i,v}$ is selected from the set $\{s(w) = e_{u,i,w} \mid i \in \{x,y,z\}, u \in N_i(v)\}$ that is most similar to s_w for the updated voxel. Since the color values are copied directly from the exemplar texture instead of using the averaged value s_w, this approach avoids the blurring problem and enables the computation of position and index histograms.

6.3.2 Search phase

The best match for each orthogonal neighborhood ($N_x(w)$, $N_y(w)$, and $N_z(w)$) of each voxel w in the output texture is found in the exemplar texture E. One way of selecting the best match is to find the pixel v from the exemplar texture that has the minimum distance between $N_i(w)$ and $N_i(v)$, where $i \in \{x,y,z\}$. The main problem in this

approach is that it is too time-consuming. Figure 6.4 illustrates the search for three orthogonal neighborhoods around a voxel.

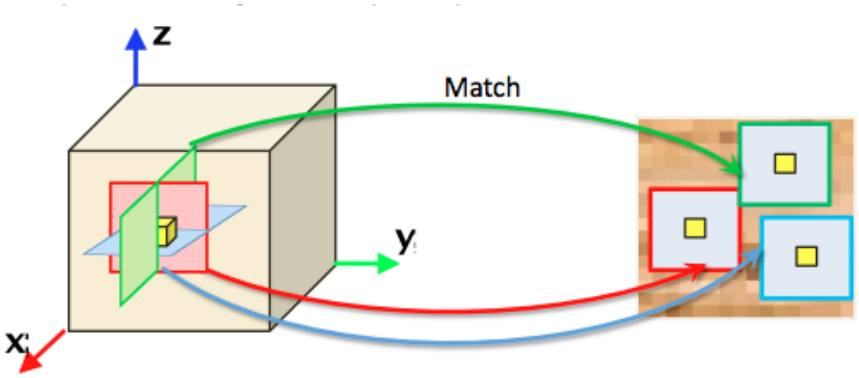

Figure 6.4: Best-matches are searched for each orthogonal neighborhood around each voxel in the synthesized texture (Kopf et al. (2007)).

The k-coherence search approach is used in order to find the best match for the corresponding neighborhoods. In the k-coherence method, for each voxel in the output texture k pixels with similar neighborhoods are found from the input exemplar texture for three orthogonal neighborhoods. This approach has an additional preprocess step and a modification for the search step:

1. Preprocess step: In the preprocess step the k-closest pixels are found for each exemplar pixel. The indices of these k-closest pixels are stored as the k-candidates (also called similarity-sets) for that exemplar pixel.

2. Restriction on search: In the search step first a candidate set is created for each output voxel by finding the union of all similarity-sets of the neighborhood pixels. Then, the best match for each voxel is found by searching this candidate set.

This approach is efficient since the number of neighborhood comparisons is restricted by the size of the candidate set.

6.3.3 Histogram matching phase

The synthesized texture should represent the statistical and structural features of the given exemplar textures. One way to achieve that is to use histogram matching techniques in the texture synthesis. The main purpose in all these techniques is to match the histogram $H_{synthesized}$ of synthesized texture to the histogram $H_{exemplar}$ of exemplar texture. The histograms can store the frequency of different values depending on the application. For example, they can store the frequency of the intensity values, or they can store the frequency of structural primitives. In Kopf et al. (2007) a color histogram matching technique is used to match the color histograms of the exemplar and the synthesized texture as closely as possible in order to retain the color information in the exemplar texture. However, this approach ignores necessary texture layout distinctions in synthesizing textures that have similar intensity values in them. To solve this problem, Chen & Wang (2010) introduced position and index histograms. The purpose of using histogram matching on both position and index is to make sure that most of the pixels in the exemplar texture appear equiprobably in the synthesized texture. The index histogram stores the frequency of each pixel, in each of the planar exemplars, being initially selected as contributing to the ultimate 3D intensity that is assigned to the target voxel. The position histogram stores the frequency of each pixel, in each planar exemplar, being closest in intensity to that ultimate assigned 3D intensity.

In this dissertation position and index histogram matching techniques are added to the MGTS framework and extended for synthesizing progressively-variant solid textures. In addition, texton histogram matching is proposed for manipulating the number of the specific textons in the synthesized textures. In this section I briefly describe the histograms (position and index) that are integrated into the MGTS framework, and I present the texton histogram proposed in this dissertation.

6.3.3.1 Histogram types

1. **Position Histogram:** This histogram is defined as a 2D image of the same size as the exemplar. It is only stored for the synthesized texture. It is computed in the copy phase of texture synthesis. When the intensity value for a voxel is computed and copied to the synthesized (output) texture, the pixel that has closest match to that intensity value is found in the exemplar texture and the frequency of that pixel in the position histogram is increased by one. The higher the frequency of a pixel, the higher the brightness of that region in the histogram. If the region is black, this means that the corresponding part in the exemplar does not appear in the synthesized texture. If the intensity values in the position histogram have similar values, this means that almost all the pixels in the exemplar texture can be found with similar frequency in the synthesized texture.

2. **Index Histogram:** This histogram is also stored only for the synthesized texture, and it is represented as a 2D image of the same size as the exemplar. It is computed in the search phase of texture synthesis. When the best matched pixels are found from the exemplar texture for each neighborhood of each voxel in the synthesized (output) texture, the frequencies of these pixels in the position histogram are increased by one.

3. **Texton Histogram:** This histogram is used for structural textures that have structural primitives (textons) in them. This histogram stores the frequency of each texton in the synthesized texture. The purpose of this histogram is to control the number of specific textons in the synthesized texture. It is stored for the synthesized texture.

6.3.3.2 Position, index and texton histogram matching

The histogram matching can be applied at different phases of the synthesis process, either at the search phase or at the copy phase. In the index and texton histogram matching, the distance in the neighborhood search is manipulated based on the match between the specified histograms in the search phase of the approach. In the position histogram matching, it is used in the copy phase of the approach. As is shown in Equation 6.2, weights $weight_{u,i,w}$ are needed to compute the color value of the synthesized voxel w at the copy phase. The modified weights are used as a means to match the histograms of the exemplar and synthesized textures.

In this section, the details of the computation of the modified weights $weight^{'}_{u,i,w}$ are explained.

1. **Position histogram matching:** To achieve position histogram matching, the histograms of the source position information for all the synthesized voxels are kept during texture synthesis. Then a re-weighting scheme is applied in the optimization (copy) phase of the approach:

$$weight^{'}_{u,i,v} = \frac{weight_{u,i,w}}{1 + max[0, H^{\text{pos}}(p(e_{u,i,w})) - \theta)]} \tag{6.3}$$

where $p(e_{u,i,w})$ refers to the position of the pixel $e_{u,i,w}$, H^{pos} is the position histogram, $H^{\text{pos}}(p)$ is the value of the position p in the histogram H^{pos}, and θ is the histogram value when all the pixels from the exemplar completely evenly appear in the synthesized texture. This equation means that if $H^{\text{pos}}(p) > \theta$, the weight is reduced to make it less likely to select p for the voxel w. If $H^{\text{pos}}(p) < \theta$, the weight is increased to make it more likely to choose $\mathbf{e}_{u,i,w}$ for the synthesized voxel w. Since in this dissertation the focus is on synthesizing a region-specific solid texture inside a model as opposed to synthesizing an isotropic solid texture inside a cube, the value of θ is computed depending on the type of the synthesized texture and

the number of voxels inside it:

- Computation of θ for isotropic textures: If the synthesized texture is an isotropic solid texture, θ is computed as the number of voxels in the model divided by the number of pixels in the exemplar texture.
- Computation of θ for region-specific textures: If the synthesized texture is a region-specific solid texture, θ is computed for each region separately. In that case, θ for a specific region is computed as the number of voxels in that region divided by the number of pixels in the exemplar texture for that region.

2. **Index histogram matching:** As opposed to the position histogram matching which is applied in the optimization phase of the approach, index histogram matching is used in the search phase to restrict the nearest neighborhood search. During the search phase, the distance between two neighborhoods is modified based on following equation:

$$d = weight_d \cdot \|s_{w,i} - e_{w,i}\|^2 \tag{6.4}$$

in which $weight_d$ is

$$weight_d = 1 + max[0, H^{\text{index}}(i(e_{w,i})) - \phi] \tag{6.5}$$

where $i(e_{w,i})$ refers to the nearest neighborhood index corresponding to the pixel $e_{w,i}$, H^{index} is the index histogram, $H^{\text{index}}(i)$ is the histogram value of the index i, and ϕ is the histogram value when all the indices completely equally distribute in the exemplar. This equation means that when $H^{\text{index}}(i(e_{w,i})) > \phi$, the distance d between two neighborhoods $e_{w,i}$ and $s_{w,i}$ will increase, making it less likely to choose $e_{w,i}$ as the nearest neighborhood index for $s_{w,i}$. Without this modification in the distance function, the search phase can converge to the same position in the exemplar, making the preservation of the texture structures impossible.

3. **Texton histogram adjusting:** Using the texton histogram matching technique, the frequency of the textons can be increased or decreased in the synthesized texture. Texton histogram matching has an opposite effect from position histogram and index histogram matching. While in position and index histogram matching the aim is to make the pixels in the exemplar texture appear uniformly in the synthesized texture, in texton histogram matching the aim is to increase/decrease the frequency of some of the textons. The decision as to the frequency of textons can be determined depending on the specific organ or region inside the organ. As in the index histogram matching technique, in texton histogram matching during the search phase the distance between two neighborhoods is modified based on the following equation:

$$d = weight_d \cdot \|s_{w,i} - e_{w,i}\|^2 \tag{6.6}$$

in which $weight_d$ is

$$weight_d = 1 + max[0, H_{\mathrm{s}}^{\mathrm{texton}}(i(e_{w,i})) - H_{\mathrm{e}}^{\mathrm{texton}}(i(e_{w,i}))] \tag{6.7}$$

where $i(e_{w,i})$ refers to the nearest neighborhood index corresponding to the pixel $e_{w,i}$, $H_{\mathrm{s}}^{\mathrm{texton}}$ is the texton histogram of the synthesized texture, $H_{\mathrm{s}}^{\mathrm{texton}}(i)$ is the texton histogram value of the index i at the synthesized texture, $H_{\mathrm{e}}^{\mathrm{texton}}$ is the texton histogram of the exemplar texture, and $H_{\mathrm{e}}^{\mathrm{texton}}(i)$ is the texton histogram value of the index i at the exemplar texture. The equation for texton histogram matching is very similar to the equation for index histogram matching except for the usage of texton histogram of the exemplar texture ($H_{\mathrm{e}}^{\mathrm{texton}}$). $H_{\mathrm{e}}^{\mathrm{texton}}$ is precomputed using the binary image of the texton (Figure 6.2-(b)). For each pixel p outside the textons, ϕ, which is the histogram value when all the indices completely equally distribute in the exemplar, is assigned to that pixel in $H_{\mathrm{e}}^{\mathrm{texton}}$. For each pixel inside the textons, the value is computed based on whether the

application needs to increase or decrease the number of textons in the synthesized texture. If the goal is to increase the number of textons in the synthesized texture, then ϕ is multiplied by a large number. If the goal is to decrease the number of textons and increase the distance between the textons in the synthesized texture, then ϕ is multiplied by a small number.

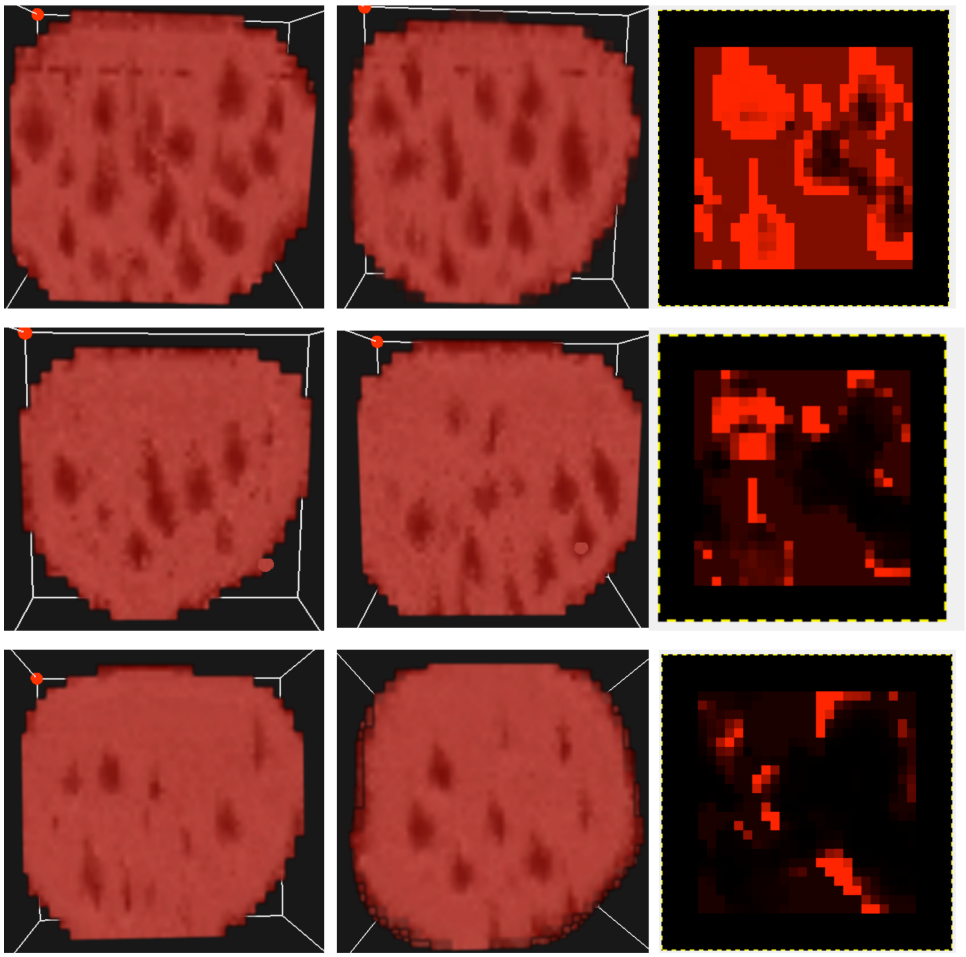

Figure 6.5: In each column, the images in the left and in the middle columns show slices of the synthesized texture, and the images in the right column show the texton histogram for that synthesized texture. For each row, results for different $H_{\mathrm{e}}^{\mathrm{texton}}$ are illustrated. The top row shows the results when the values inside the texton regions are set to a high value in $H_{\mathrm{e}}^{\mathrm{texton}}$. The other rows show the results when these values are set to successively lower values.

Figure 6.5 shows the results obtained using the texton matching method. As is shown, this method can be used for the illustration of hyperplasia (proliferation of cells

within an organ or tissue) in anatomical organs.

Discussion of histogram matching

Color histogram matching, proposed in Kopf et al. (2007), is not always an appropriate technique since it only works for color but not for structural features available in the texture. In addition, when the texture channels are decorrelated it fails to preserve the color histograms. For the purpose of this dissertation, it is not appropriate for most of the anatomical textures, especially when the colors of the texture structures are similar to each other.

Using position and index histograms, both texture structures and color histograms can be preserved in the synthesized textures. Position histogram matching ensures that color histogram matching is also satisfied, since position histogram matching considers that all the pixels in the exemplar should have the same probability to appear in the result, which would preserve the color histograms and texture structures in the synthesized texture. The purpose of these matching techniques is to make sure that every pixel in the exemplar texture will appear same number of times in the synthesized texture. These techniques are really useful if the aim is to have the same local and global statistics in both exemplar and synthesized textures. However, if we want to change the number of specific textons in the texture, then texton matching is a useful technique. It gives the user the ability to control the amount of specific textons in the synthesized texture. This can be used for visualizing hyperplasia (proliferation of cells within an organ or tissue) in an organ.

6.3.4 Multi-resolution

As in most texture synthesis approaches, the MGTS framework uses multi-resolution. It generates the output texture from the lowest resolution (coarse version of volume) to the highest resolution (finer version of volume). It uses trilinear interpolation to switch from a coarse level to a finer level. There are two main advantages of using multi-resolution

in texture synthesis:

1. It decreases the computation time required to synthesize the solid texture.

2. It increases the quality of the synthesized textures by preserving the relations between the texture primitives.

Figure 6.7 show the effect of multi-resolution in the synthesized texture. The exemplar texture in Figure 6.6 is used for synthesizing the solid textures.

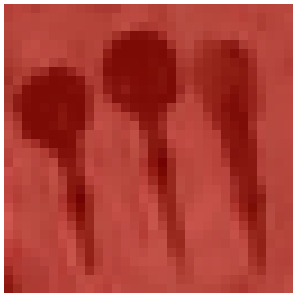

Figure 6.6: 2D exemplar texture for illustrating glands.

Figure 6.7: (Top row) slices of solid texture synthesized without multi-resolution. (Bottom row) slices of solid texture synthesized with multi-resolution.

6.4 Vector-guided Solid Texture Synthesis

In order to synthesize solid textures that are oriented along the model-based vector field, the neighborhood search phase of the synthesis algorithm is extended. In this phase of the approach the orthogonal neighborhoods $s_{w,i}$ $(i = x, y, z)$ are rotated in 3D space around their center by a rotation matrix R_w, which is computed by considering the vectors du_w, dv_w, and $d\tau_w$ at that voxel w.

$$R_w = \begin{bmatrix} du_w^x & dv_w^x & d\tau_w^x \\ du_w^y & dv_w^y & d\tau_w^y \\ du_w^z & dv_w^z & d\tau_w^z \end{bmatrix}$$

$$sr_{w,i} = R_w * s_{w,i}$$

Here, the rotated neighborhoods are denoted by $sr_{w,i}$ $(i = x, y, z)$. In our case, the pixels in $sr_{w,i}$ $(i = x, y, z)$ are resampled from that of synthesized texture s. Figure 6.8 illustrates the three rotated neighborhoods of a selected voxel in the synthesized texture and the best-matches found for them in the exemplar texture.

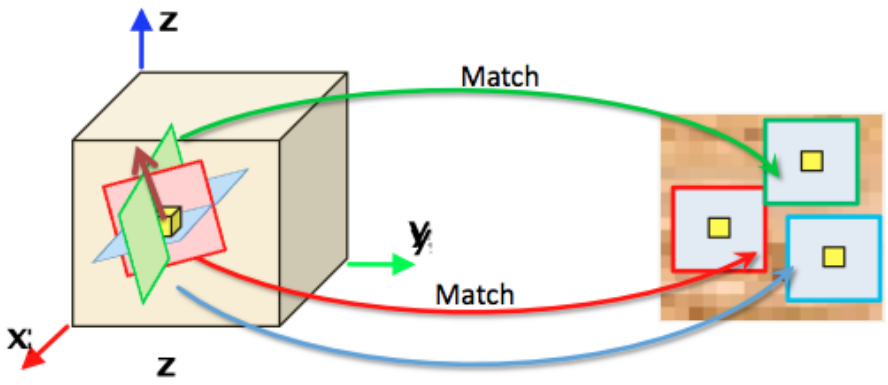

Figure 6.8: Best-match is searched for the three oriented neighborhoods in the synthesized texture.

The approach proposed in this dissertation for synthesizing oriented solid textures is an improvement over the method presented in Chen et al. (2009) since the misalignment

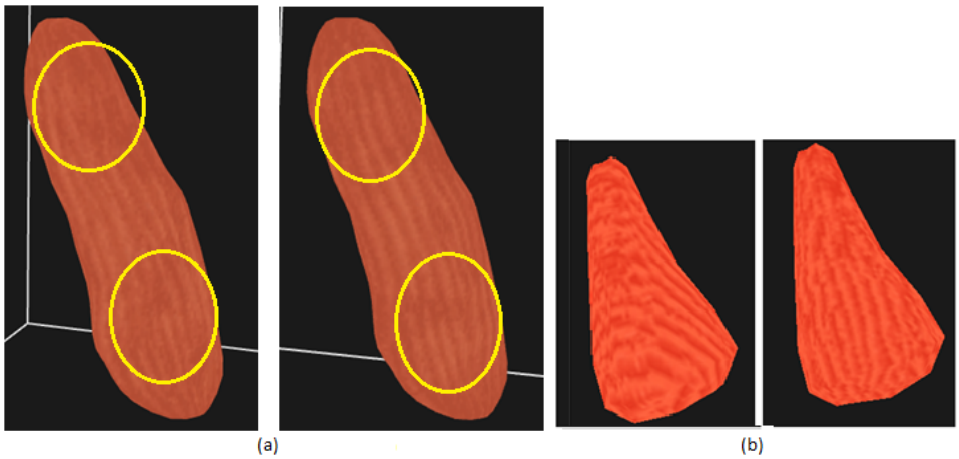

Figure 6.9: Comparison for solid texture synthesis: (a) solid textures synthesized for the scm by my method. Neighborhoods are rotated by considering left: only du; right: both du and dv. (b) Comparison of Chen et al. (2009)'s approach with my method: left: Chen et al. (2009)'s approach; right: my approach

of the structural feature of the texture along the vector field is avoided by rotating the neighborhoods in 3D space using the rotation matrix that is computed by considering both du and dv. Figure 6.9a illustrates that using both du and dv in the computation of the rotation matrix improves the result. In Chen et al. (2009) the neighborhood for each plane is rotated in 2D space by considering the projection of the orientation vector on that plane (Figure 6.10). Using this method, the continuity of the structures cannot be preserved and the structural features of texture may not follow the guidance vector field. However, in the new approach the continuities can be preserved since the neighborhoods are rotated by considering model-based smooth vector fields. Figure 6.9.b illustrates the effects of the proposed algorithm compared to the approach presented in Chen et al. (2009). For both illustrations the same 2D examplar textures are used in texture synthesis. For the first illustration only the along-object vector du_w is considered in texture synthesis for rotating the neighborhoods using the approach presented in Chen et al. (2009). In the second illustration the results are obtained using the proposed method in this dissertation.

The synthesis order of the voxels has a significant effect on the output texture's qual-

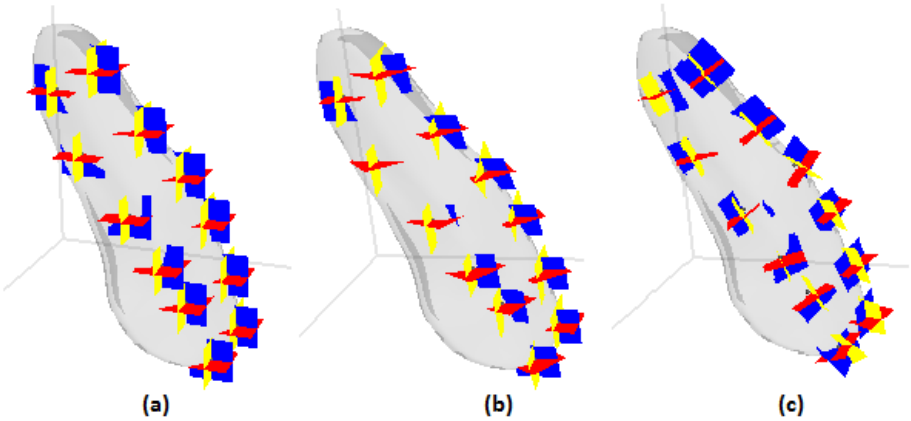

Figure 6.10: Comparison of neighboorhoods in solid texture synthesis: (a) orthogonal neighborhoods; (b) neighborhoods in Chen et al. (2009); (c) neighborhoods computed using our approach

ity. Most algorithms such as Kopf et al. (2007) grow a synthesized patch by selecting the voxels randomly or sequentially in the search phase. That approach is not appropriate in oriented solid texture synthesis since the structural features of the textures may not be preserved along the orientation field. To solve this problem, I adapted the oriented surface texture synthesis method so that it follows the streamlines on the surface (Zhang et al. (2003)) for oriented solid texture synthesis. I first computed streamlines from the 3D vector files inside the volume. Then I synthesized the texture by considering the individual streamlines to decide the order of voxels for the neighborhood search.

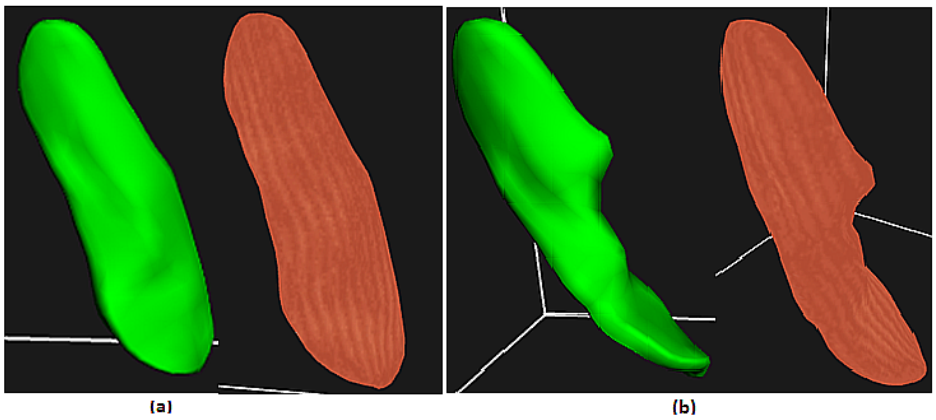

Figure 6.11: Solid textures synthesized for two different scm models.

The proposed approach is consistent for different instances of the same model. By

this method two different scm models with the same medial representation structure are textured automatically (Figure 6.11).

6.5 Material-guided Solid Texture Synthesis

Progressively variant solid textures need to be synthesized for anatomical organs such as the mandible and the scm. For example, inside the mandible there are different tissue types along depth. On the other hand, the scm has homogeneous tissue strands inside it but attaches to the bone at each end by tendons, which look like white muscle fibers.

In order to imitate these features in the patient-specific illustrations, the oriented solid texture synthesis method outlined above was further extended by considering model-specific material constraints in choosing the exemplar texture region that is going to be used during texture synthesis. As in Zhang et al. (2003), a 2D exemplar texture is used that varies in color or material. Each exemplar texture has a feature image ef and a transition image et. The feature image stores the structural properties of the textures, such as a signed distance field obtained from a binary mask of the image, and the transition image stores the transition of color or material information in the texture. These textures can be either designed using imaging tools or other image processing techniques (Zhang et al. (2003)).

In the neighborhood search and optimization phase of the solid texture synthesis, only the exemplar pixels that satisfy the condition given in Equation 6.8 are considered for computing the color value in voxel w.

$$||\mathbf{st}_{w,i} - \mathbf{et}_{w,i}|| < threshold \tag{6.8}$$

Figure 6.12 illustrates the neighborhood search and selection process for the three neighborhoods of a voxel by considering the material transition of the exemplar texture and the synthesized texture. For simplicity, the process is shown for a cubic texture

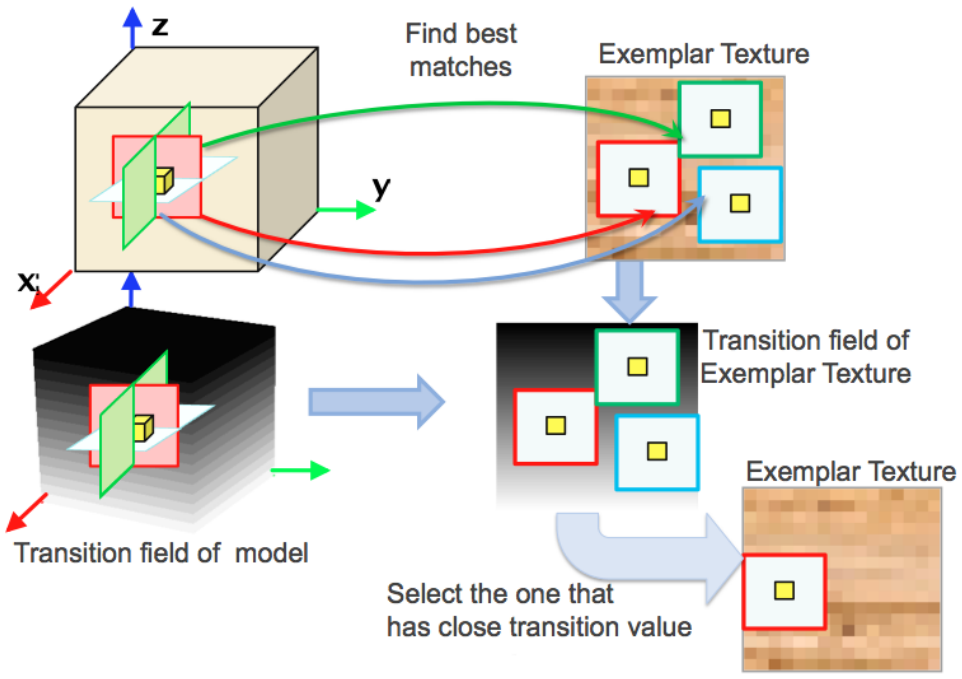

Figure 6.12: Best-match is searched and selected using the the material transition information.

and the neighborhoods are shown orthogonal to each other. For material-guided texture synthesis inside a specific model, the process is achieved for a transition field inside the model using rotated neighborhoods for each voxel inside it.

Synthesizing progressively variant textures can be applied to many organ illustrations. For the illustration of the scm there are two different regions along the model. In patient-specific illustration of the mandible my objective is to obtain an illustration similar the one in Netter (2009). In the anatomic illustration of the mandible the texture elements are different for the inner layers (bone tissue) and outer layers (endosteum tissue). Both of these features are illustrated by synthesizing solid textures for a specific patient's mandible using the proposed approach (Fig. 6.13 and Fig. 6.14).

Algorithm 6.1 Guided Solid Texture Synthesis

$e_w^0 \leftarrow$ **random neighborhood in e**
$et_w^0 \leftarrow$ **random neighborhood in et**
$R_w \leftarrow$ **Compute rotation matrix using** du_w,dv_w **and** $d\tau_w$
for $n = 0$ to N **do**
  $s_w^{n+1} \leftarrow$ **arg min** $E_t(s_w; e_w^n)$
  for $i = (x, y, z)$ **do**
    $sr_{w,i}^{n+1} \leftarrow$ **Rotate** $s_{w,i}$ **using** R_w
    $str_{w,i}^{n+1} \leftarrow$ **Rotate** $st_{w,i}$ **using** R_w
    $e_{w,i}^{n+1} \leftarrow$ **Find nearest neighbor of** $sr_{w,i}^{n+1}$ **in** e_i
    which satisfies $||et_{w,i}^{n+1} - str_{w,i}^{n+1}|| < threshold$
  end for
  if $e_w^{n+1} = e_w^n$ **then**
    $s_w \leftarrow s_w^{n+1}$ **break**
  end if
end for

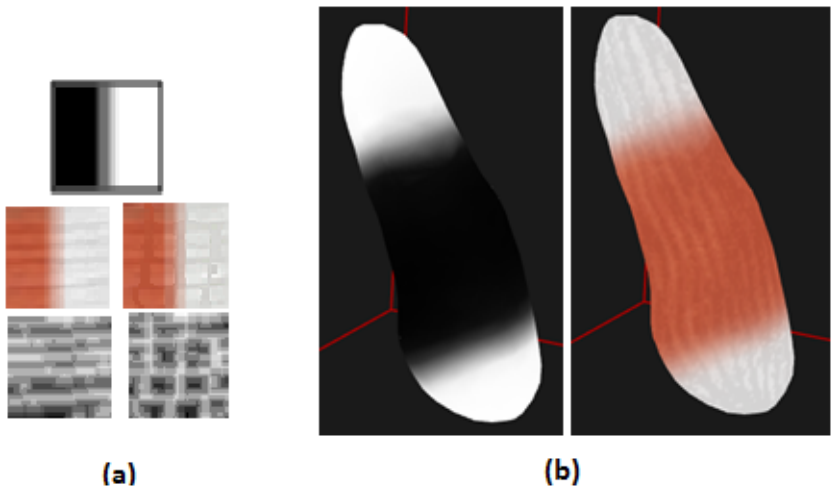

Figure 6.13: Different textures are synthesized along the model for different regions of the scm.

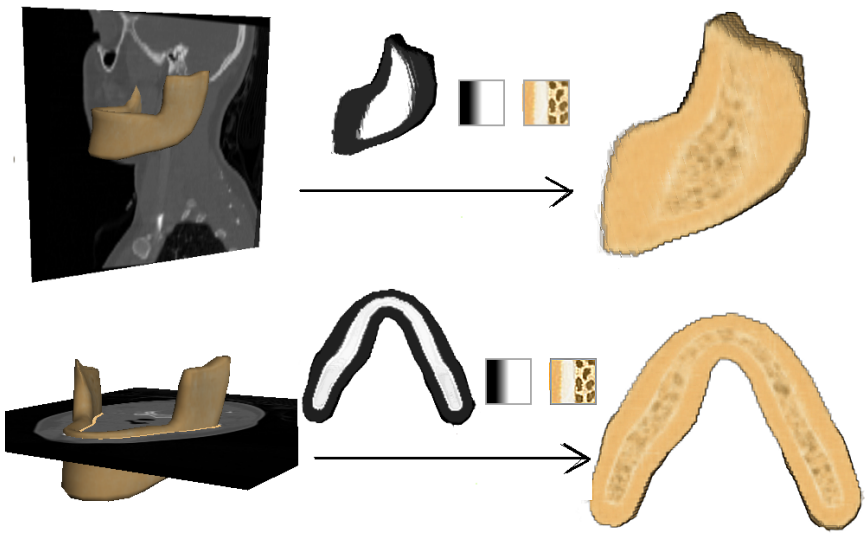

Figure 6.14: Different textures are synthesized for different regions of the mandible. (Left) clipping planes along and through the mandible; (middle) inputs to solid texture synthesis: transition field for the mandible, exemplar texture with its transition image; (right) synthesized solid texture visualized by clipping planes.

6.6 Scale-guided Solid Texture Synthesis

Scale-guided solid texture synthesis is achieved in a way similar to that in model-guided solid texture synthesis. First, the anisotropic 2D exemplar texture, which has variation in scale, is created using 2D isotropic textures and the target variation in the synthesized 2D texture.

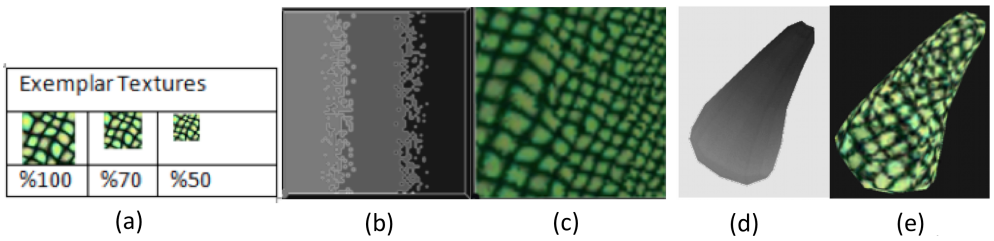

Figure 6.15: (a) 2D exemplar textures which differ in scale; (b) the transition of scale values in the anisotropic 2D exemplar texture; (c) the anisotropic 2D exemplar texture; (d) the transition of the scale values for the target solid texture on the model; (e) the scale-guided solid texture synthesized for the model.

Figure 6.15a shows the 2D exemplar textures which differ in scale, and Figure 6.15c

shows the synthesized anisotropic 2D exemplar texture. Second, this anisotropic 2D exemplar texture is used for synthesizing solid texture by considering the transition of the scale values. The approach for synthesizing material-guided solid textures is applied for synthesizing scale-guided solid textures. Figure 6.15d shows the target scale transition on the pyramid model, and Figure 6.15e shows the synthesized solid texture which has variation in scale.

6.7 Model-based Texture Synthesis Control

In anatomical illustrations such as Figure 1.6 the muscle textures look different for different clipping planes. The important thing here is that the texture look different based on the orientation of the clipping plane. It is very computationally expensive to constrain the texture on the clipping plane for every possible orientation in texture synthesis since we have to do neighborhood search for each constrained direction.

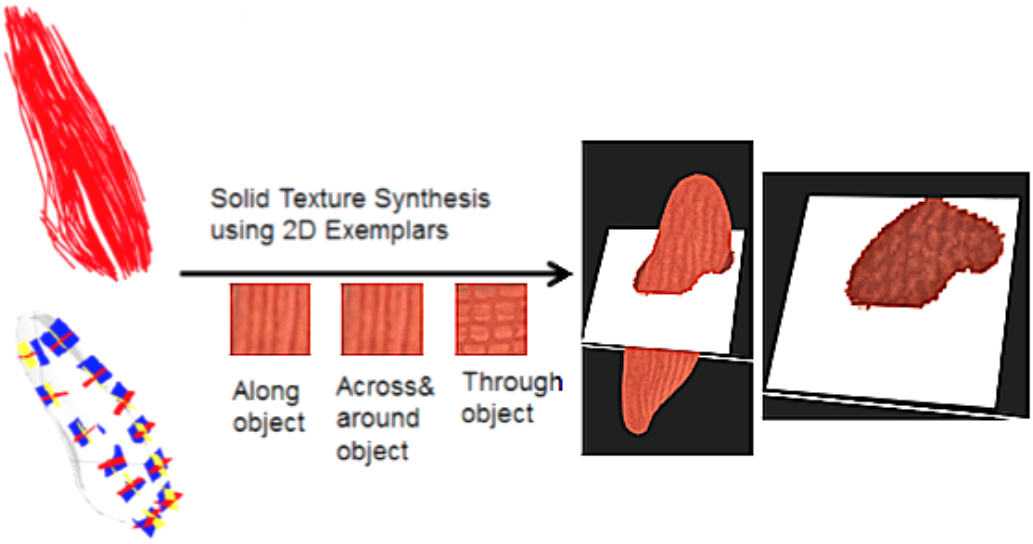

Figure 6.16: Model-based texture synthesis control is achieved using different exemplar textures for each oriented neighborhood.

In anatomical illustrations the models are mostly clipped either perpendicular to the medial axis or along the medial axis. Thus, texture synthesis is constrained to these

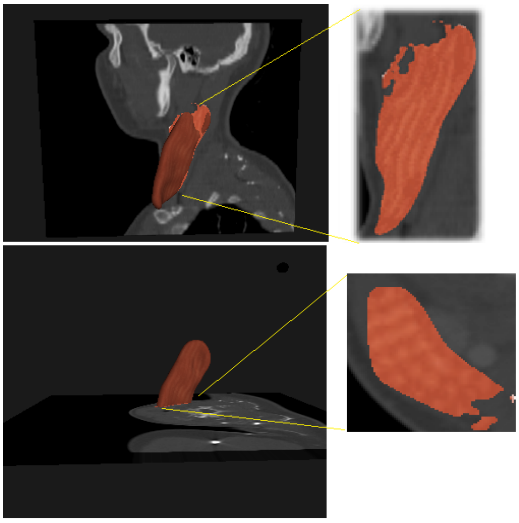

Figure 6.17: Texture appearance changes based on the orientation of the clipping plane: (top) along object clipping plane; (bottom) through object clipping plane

directions. To do so, different 2D exemplar textures are picked as input for different longitudinal vs. axial directions, and neighborhoods are rotated in the search phase of the approach as in oriented solid texture synthesis. Figure 6.16 illustrates the inputs needed for synthesizing orientation constrained textures for the scm. It also shows the textures on and inside the model. Figure 6.17 shows the textures and the CT data on the clipping planes along the model and through the model.

The selection of 2D exemplar textures in this method is very important since they should contain similar colors. Otherwise the synthesized texture may contain colors that are not in any of the exemplar textures due to the optimization phase in the method. To solve this problem, the approach presented in Welsh et al. (2002) is used to match the colors of the exemplar textures in texture synthesis. Color is transferred from one exemplar texture to the other one.

In Kopf et al. (2007) using different exemplar textures for each neighborhood direction is also used for texture synthesis control. However, as is seen in Figure 9 of Kopf et al. (2007), the circles on the cross section can form lines which is not desirable. Thus, in this dissertation I added a correction term to solve this problem. First, in the search phase the sums of the distances between the neighborhoods in each of the three orthogo-

nal directions are computed separately. Then, in the optimization phase of the approach the weights are increased by a small amount if the sum of the distances between the neighborhoods in that direction is more than the distance between the neighborhoods in the other directions. This is to give more importance to the directions that do not match very well. Figure 6.18 shows the effect of this reweighting scheme.

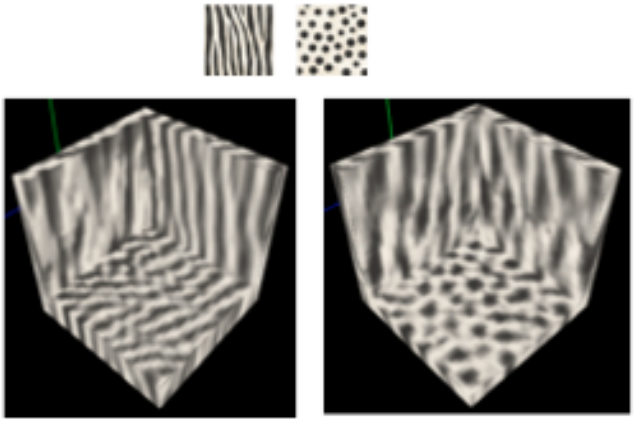

Figure 6.18: (Left) When the method in Kopf et al. (2007) is used, circles may form lines in the cross section of synthesized texture even if this direction is constrained by a circle texture. (Right) When the weights are updated based on the similarities for each neighborhood, circles on the cross section of the synthesized texture become more apparent.

6.8 Texton-based Solid Texture Synthesis

In this dissertation a texton-based solid texture synthesis approach is presented for synthesizing textures that have high structural components. In this approach the inner regions of textons are represented by an approximate 2D medial representation (Figure 6.2). The purpose in this approach is to use the texton-based coordinates in the search phase of the approach in order to control the construction of 3D textons from 2D textons.

Texton-based texture synthesis is achieved using a multi-step neighborhood search scheme. Initially when searching neighborhoods, for each voxel the k-best matching neighborhoods are collected in two orthogonal neighborhoods, specifically the neighbor-

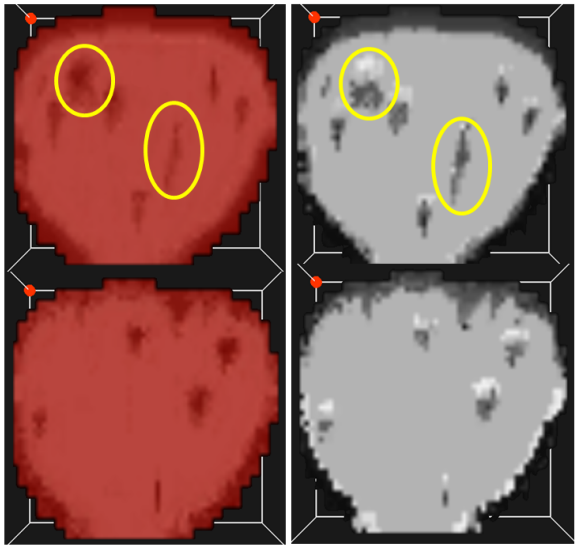

Figure 6.19: Comparison between solid textures synthesized (top) without and (bottom) with texton-based solid texture synthesis. Solid textures are synthesized using the 2D exemplar textures in Figure 6.2.

hoods x and y. The best-match for the third neighborhood z is not found since it is hard determine what will be the shape of the texton along that dimension. Among those k-best matches, the best matches are found based on the medial coordinates of textons, and these values are used for that voxel. Figure 6.19 illustrates the effect of using textons in solid texture synthesis. The top row of this figure illustrates one slice of the solid texture obtained without using texton information. As seen in the regions inside the yellow circles, the shape of the textons on that particular slice do not look like the shape of the textons in the 2D exemplar texture. On the other hand, in the bottom row of this figure it is seen that the shape of the textons are preserved when the texton information is used in the synthesis process. Though this addition improved the quality of the results for textures that have structural primitives on a uniform background (e.g., gland texture inside prostate), for simpler shape textons (e.g., stripes in muscles) no improvement was observed since basic exemplar-based texture synthesis performed quite well for those textures.

6.9 Results for Illustrative Medical Visualization

Given segmented models, which can be generated automatically using methods from image analysis, and given exemplar textures from medical textbooks and other medical illustrations, the MGTS framework is fully automatic and does not require any user interaction during either model-based part or texture synthesis part. There are several parameters in MGTS that can be set by the user, such as the thresholds for transition regions and the size of the neighborhoods.

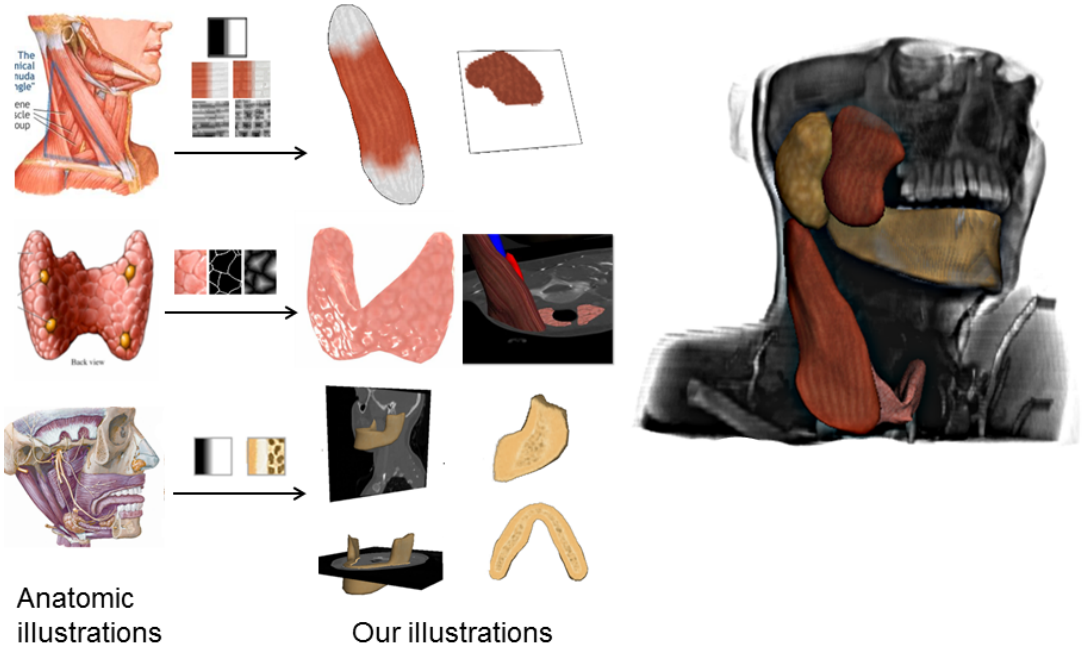

Figure 6.20: Solid textured models (the parotid, masseter, scm, mandible, thyroid) integrated into the model-guided rendering framework

As indicated in the introduction of this chapter, the goal of the MGTS framework is eventually to be able to fill a space containing many organs with synthesized solid textures to create an illustrated look for a particular patient. In Figure 6.20 solid textured models (the parotid, masseter, scm, mandible, thyroid) are visualized inside the CT data using the Model Guided Rendering (MGR) framework (Merck (2009)) to identify these organs.

While the solid textured models in a volume rendering environment help the user to

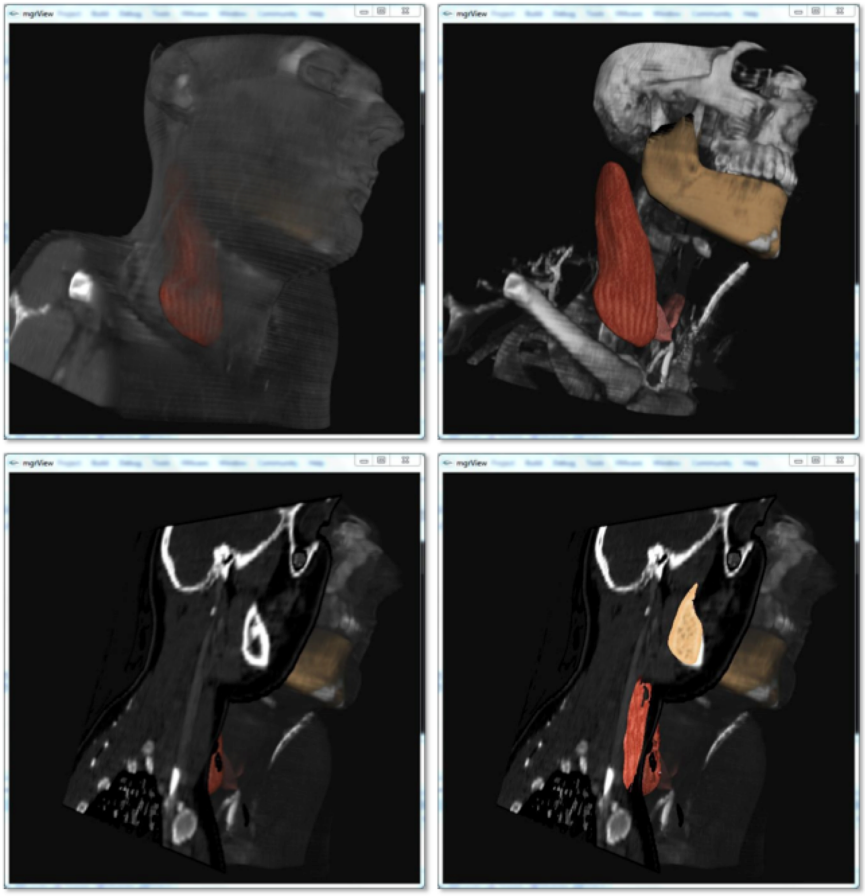

Figure 6.21: (Top row) solid textured models are visualized inside volume rendering. (Bottom row) a clipping plane is applied to the 3D data. The left image illustrates that without textures it is hard to distinguish the anatomical organs and the regions inside them. The right image shows that with solid textures mapped onto the 2D slices, the organs can be easily identified.

distinguish the organs inside the CT data, the textures mapped onto a 2D slice help the user to distinguish them in a 2D slice view of CT data. In Figure 6.21 solid textures are mapped onto a 2D slice to visualize and distinguish the organs on that slice.

Figure 6.22 demonstrates the usage of texture in radiation dose visualization. Here the textures help identification of organs whose surfaces are washed with dose colors. For the applications in which color is used to encode a separate data set, like dose in radiation treatment planning, texture can be used to identify the models and their shapes.

The MGTS framework is not designed for illustrating pathological regions inside

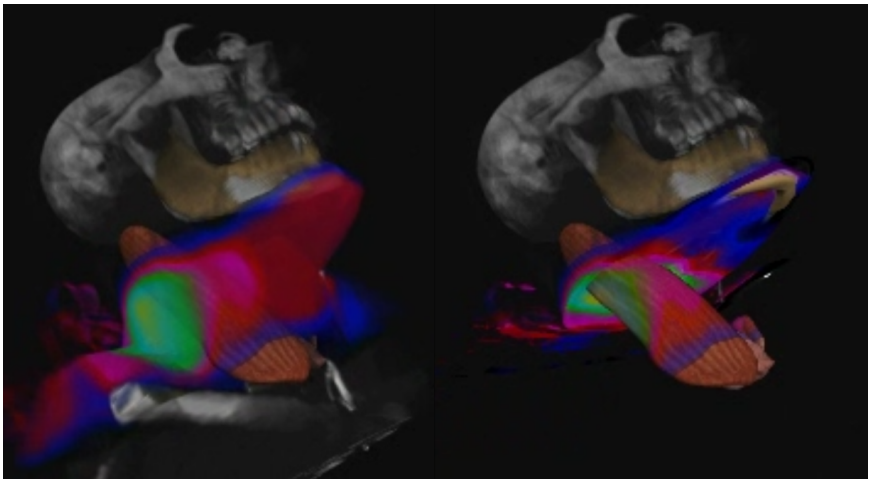

Figure 6.22: (Left) Solid textured models under dose color wash in 3D. (Right) Dose is mapped onto the textured model.

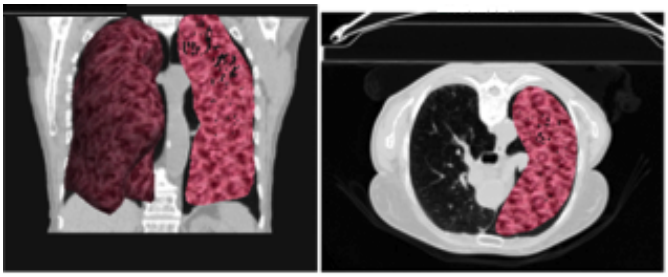

Figure 6.23: Solid textured lung model. The texture is synthesized for illustrating the emphysema of the lungs assuming that it has this disease. Sample illustration is available in "http://www.virtua.org/"

an organ since image intensity information inside the organ should be investigated for determining these regions. However, the framework has capability of synthesizing textures that express a pathology (e.g., sickness) in the organ. In Figure 6.23 the textbook illustration on the top shows the emphysema of the lungs for a typical patient. A 2D exemplar texture is cropped from this illustration and used for synthesizing lung texture for a real patient's lung model. The synthesized textures on and inside the model are shown in the two images below the illustration.

The MGTS framework can also be used for synthesizing complex objects that consists of many substructures, such as the kidney. Figure 6.24 illustrates some of the textured structures inside a patient's kidney.

As shown with these results MGTS has advantages over other approaches (Chen

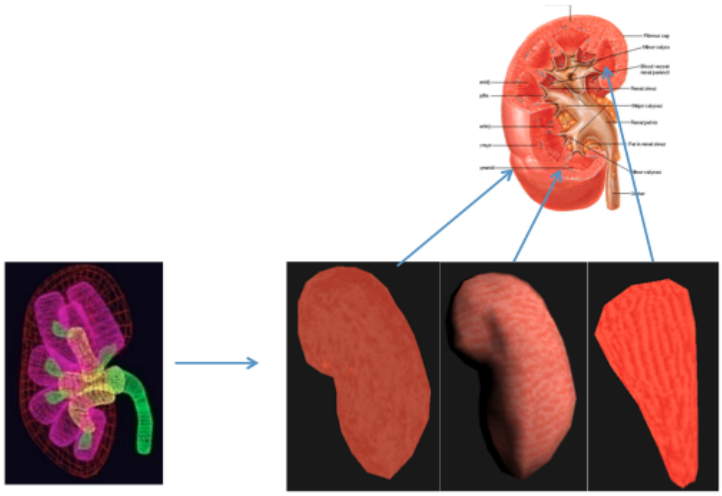

Figure 6.24: Solid textures are synthesized for the layers of kidney and for the pyramids inside the kidney.

et al. (2009)) because it can generate consistent and detailed progressively changing solid textures from 2D exemplar textures. Different solid textures can be produced for different models using the proposed framework, and they can easily be integrated with the CT image data. In addition, the illustrations can be used to highlight the models inside the CT data for different application. The drawback of the framework, however, is the cost in both computation and memory, as it explicitly computes and stores a dense 3D array of voxels covering the entire target model. In addition, the guidance information is based on the model, not the image-intensity pattern. In the future these drawbacks can be dealt with, and new approaches can be added to the system.

Chapter 7

Conclusion

Traditional medical illustration has focused on presenting information in an effective, efficient and attractive way to understand the anatomy of a typical patient. Its main goals are to record and disseminate medical knowledge by reducing visual overload. Computer-supported medical illustrations use different visual elements, such as shading, cutting, deformation, and annotation to achieve that goal.

While in scientific visualization the objective is to provide detailed, objective information by mapping data to color value, in illustrative visualization the goal is to provide aggregated or abstracted information; sometimes subjective (e.g., personal ideas / theories) by mapping data to colors, textures, symbols, styles, patterns, sketches, etc. The focus of this dissertation is to provide one of these means, texture, to obtain illustrative visualizations of anatomic objects. The main goal of the model-guided texture synthesis (MGTS) framework is to maintain the global comprehension from traditional medical illustrations while visualizing the shape of the anatomical structures of individual patient data. There are methods that try to create illustrative visualization of patient data, such as methods in Lu & Ebert (2005) which use a color-transfer function for isotropic texture cubes. However, none of these methods consider the variation of materials inside the anatomical models.

The main advantage of MGTS over the other methods is that it considers the guid-

ance information obtained from model-specific parameterization and region-specific textures in order to synthesize region-specific anisotropic textures. Texture generation requires only limited manual intervention, which consists of setting some parameters. MGTS is designed to be based on deformable model-based segmentation technologies, which have been under development at UNC. It can be applied to a broad variety of patient-specific, segmented anatomical models. One can interact with it (e.g., using model-based clipping planes to see the interior regions of a textured organ) in a 3D visualization environment, for example in the Model Guided Rendering framework proposed in Merck (2009).

This chapter has four main sections. First, there is a review of the dissertation's thesis statement and claims in the context of the methods and materials presented in the previous chapters. Second, there is a discussion of some of the problems of the methods and framework presented along with some limitations of my study. Third, there are suggestions of directions for future research and for method improvements. Fourth, there are implementation details about each component of MGTS.

7.1 Thesis Statement and Claims Revisited

In this section, I revisit the contributions and the thesis that were presented in Section 1.4 to summarize how each one was addressed.

This dissertation supports the following thesis:

Combining texture synthesis (in 2D and 3D) with volume parameterization provides a framework for generating illustration of patient-specific data. Also, structural and material variations of the organs can be illustrated by textures obtained using a metamorphosis approach that takes intensity and structure into account to generate a smooth transition from one texture to another.

There are two parts to the demonstration of this thesis. First, the methodology

claims have been presented in Chapters 3, 4, 5, and 6. These methods are texture metamorphosis and model-based guidance generation methods for creating the inputs of the framework, and surface and solid texture synthesis methods for generating model-specific textures using these inputs. The contribution of texture metamorphosis is discussed in contribution number 2 below. The contribution of model-based guidance generation is discussed in contribution number 1 below. The contributions in texture synthesis are discussed in contributions numbered 3 and 4. The contributions in model-based, material-specific texture synthesis are discussed in contribution number 5 below.

Second, there are the results obtained by these methods that have been presented in these sections. Texture metamorphosis has been applied to different textures that have different complexities, and the results along with comparisons have been presented in Section 3.7. Model-based guidance information is created for different anatomical models that have medial representations, such as along-object vector fields and material transition information for different instances of the scm presented through Chapter 4. Surface and solid textures synthesized for different anatomical models have been presented in Section 5.4 and Section 6.9. The contributions related to the demonstration of the framework have been presented in contribution numbers 6 and 7 below.

The contributions of this dissertation are as follows:

1. ***A method of computing 3D model-based guidance information, such as an orientation field and a material transition field, for different instances of the same anatomical organs using a volumetric model-based coordinate system obtained from a medial representation.***

   Guidance information, such as orientation and material transition field, for a specific model is one input of texture synthesis for creating anisotropic, model-based textures. Though some applications use sketch-based techniques for creating this information, this process is too time-consuming and may not be consistent for different instances of the same model. Chapter 4 presented a new method for ob-

taining model-based guidance information using medial representations. One can obtain consistent orientation and material transition fields for different instances of various anatomical models by extending the along-, across-, through- (X2U) map proposed in Merck (2009). This approach does not require any user intervention.

2. ***A method of generating progressively changing 2D exemplar textures using a new energy-based metamorphosis approach. This method utilizes appearance information along with the structural features in a single framework to create transformations from a source texture to a target texture.***

   The synthesis of progressively-variant textures requires progressively-variant 2D exemplar textures as input. As with existing methods, my method does this via texture interpolation. However, my metamorphosis method overcomes the weaknesses of the interpolations in the existing methods (see Figure 3.9 and Figure 3.10), which are based on the warping of the structural features of the textures without considering the transition of the intensity values.

   The optimization-based texture metamorphosis approach for texture interpolation, presented in Chapter 3, is a new energy optimization scheme derived from optimal control principles that exploits the structure of the metamorphosis optimality conditions. It considers changes in pixel position and pixel appearance in a single framework. In contrast to previous techniques that compute a global warping based on feature masks of textures and use texture synthesis to generate the morphed textures, the approach allows transforming one texture into another by considering both intensity values and structural features simultaneously. The approach presented in this dissertation can morph stochastic, semi-structural and regular textures, with different levels of complexity. Results along with the comparison with their methods have been presented in Section 3.7.

3. ***A method for constraining the synthesis of structural primitives in a solid texture using medial-based representations of the structural primitives (textons). The solid textures are synthesized from three 2D exemplar textures. The medial representations are generated from the 2D exemplar textures, and they are used in forming the synthesized solid texture.***

   Because the original exemplar-based solid texture synthesis algorithm lacks shape information of structural primitives (textons) in exemplar textures, it performs poorly in synthesizing these structural primitives in the output solid texture. Parameterization of the textons provides the additional information for constraining the selection of values from the exemplar textures in texture synthesis. In Section 6.8 I showed that medial axis parameterization of the textons along with their labels improves the quality of the synthesized textures, especially for textures that have structural primitives on a uniform background (e.g., gland texture inside the prostate). For simpler shape textons, such as stripes in muscles, no improvement was observed since basic exemplar-based texture synthesis performs quite well for these textures.

4. ***A method for controlling the number of textons in a texture synthesized from exemplars using a texton-based histogram matching technique in texture synthesis.***

   In most texture synthesis approaches the purpose is to maintain the features in the exemplar texture as much as possible in the synthesized texture. The recent work on position and index histogram matching techniques is effective for synthesizing textures that contain the values from the exemplar texture uniformly. For example, structural textures that have same number of textons as the exemplar texture can be created via these techniques. Controlling the numbers of the

textons in the synthesized texture can be beneficial for illustrating different cases, such as hyperplasia in specific organs. In Section 6.3.3 I introduced a new method, texton-based histogram matching, for manipulating the number of textons in the synthesized texture. The results were illustrated for prostate textures.

5. ***A method of synthesizing progressively-varying, anisotropic, model-based surface and solid textures for obtaining "Netterly" renderings of patient-specific anatomical organs using a novel framework. This framework uses model-based guidance information and progressively changing 2D exemplar textures as input and creates textured anatomical models as output. Depending on the application, the contributions in item 3 and 4 are used in texture synthesis.***

   When synthesizing isotropic, uniform textures from a 2D exemplar texture, it is assumed that the output texture will have the same orientation, scale, and material features as in the given exemplar texture. On the other hand, when synthesizing anisotropic, region-specific textures, the output texture should be synthesized based on a guidance information using interpolated exemplar textures as input. Specifically, the guidance information specifies how the values in the output texture will be selected from the interpolated exemplar textures. It contains the variation of a particular information (e.g., orientation, scale, material type) in the output texture. As indicated in item 1, in this dissertation this guidance information comes from model-based volumetric parameterization. The second input, region-specific interpolated exemplar textures, is used to create the output textures under the guidance information. For instance, if material transition is given as the guidance information, the interpolated textures should contain variation in their material characteristics. My contributions of methods for creating the interpolated textures were discussed in item 2.

As presented in Chapter 6, using the model-based guidance information on and inside the model and 2D interpolated exemplar textures as input, the MGTS framework synthesizes textures on and inside the model that change progressively in orientation and material according to the given guidance information. Depending on the application and texture type, the synthesis process can take advantage of the structural texture primitives and their parameterization. Contributions related with these were discussed in contribution numbers 3 and 4. In the results section of Chapter 5 and Chapter 6, the robustness of this method was illustrated with a variety of textures applied to different anatomical structures, such as the muscles and the mandible.

6. ***Demonstrating that MGTS supports the illustration of any patient's segmented data so as to achieve illustrative visualization of his/her organs. This is useful to identify and convey the shape of the organs in a complex environment with many structures.***

   This claim summarizes my thesis and has been dealt with throughout this dissertation. In particular, surface and solid texture synthesis methods, both of which allow synthesizing patient-specific textures for the anatomical models, are used to achieve this goal. These algorithms were applied to a variety of anatomical organs (e.g., the mandible, the scm, the thyroid) using a variety of textures (stochastic, semi-structural, structural, static, non-static). In all these textured models, the objective was to obtain illustrative rendering of the anatomical organs using textures along with their stochastic and structural features for improving anatomical understanding of the acquired data. The results were presented in Section 5.4 and Section 6.9.

7. ***Demonstrating the usage of texture in radiation dose visualization to help identification of organs whose surfaces are washed with dose***

colors.

For the applications in which color is used to encode a procedural information, like dose in radiation treatment planning, texture can be used to identify the models and their shapes. In Chapter 6.9 the textured models are used for identifying the objects in a complex environment under the dose.

7.2 Limitations

In this section limitations of the MGTS framework along with the limitations of the related methods used in it are discussed:

1. Abstraction degree via textures: Illustrative textures in medical illustrations are being used for depicting the characteristics of organs within the human body, while at the same time removing details that are not necessary for understanding their material characteristics, shape and functions. On the other hand, textures obtained with photography and modern scanning methods provide too many details and too complex lighting operations that can decrease the understanding of the essential information in the organs. Thus, the kind of textures to be used is a decision that should be made depending on the degree of abstraction wanted in a particular application. It is essential to adapt the appearance of objects or regions to their relevance for a specific application. For instance, for a surgical simulation, photographic textures are more appropriate since this trains the users for interacting with the real data in the future. Though in this dissertation I used the textures that are not realistic, the methods utilized in this dissertation can be extended to realistic textures.

2. Sources of medical illustrative textures: Due to the lack of a standard database for medical textures, the textures used as exemplar textures in the MGTS framework have been cropped from textbook medical illustrations. However, these cropped

exemplars include variations due shading and lighting affects used in the illustrations. This causes problems in texture synthesis. This work tried to remove these variations using imaging toolboxes as a means of creating the exemplar images.

Though my results are preliminary for a clinical application, controlled user studies can be done with doctors and medical illustrators for evaluating my system. In this study it could be investigated whether the addition of textures help identification of the organs.

3. Dependency on segmentation: Segmentation of anatomical organs is a challenging task. Careful segmentation is important for many applications, such as radiation treatment. Minimally interactive solutions using different approaches is an area of ongoing research. In this dissertation it is assumed that the segmentation of all relevant structures with their medial representations are available for synthesizing textures specific to that segmented model.

The methodologies included in the MGTS framework also have some limitations:

1. Selection of texture features: Texture features, which were presented in detail in Section 2.1.2, are essential for representing the content of the textures. Different texture features are selected depending on the purpose of the technique and the type of the texture (e.g., stochastic, semi-structural, structural). In both metamorphosis and synthesis, texture features are important for preserving and propagating the stochastic and structural details of the textures. Structural texture features, such as texton masks and signed distance fields based on this masks, are used in texture synthesis and metamorphosis in the MGTS framework. The process of deriving texture features is not done automatically in every texture. For instance, the textons are specified by the user using an imaging toolbox. Since texture analysis and derivation of texture features is not the focus of this dissertation, available techniques were used for achieving that.

2. Preservation of texture content: In texture synthesis the details of the textures can be lost due to the averaging in optimization or to lack of control on global statistics in the best neighborhood search. Even when the texture is invariant, preserving its features is a hard problem by itself. When the texture has many overlapping structural primitives, this problem becomes much harder. This is one of the limitations of my approach, which I tried to avoid by using additional feature channels in texture synthesis. However, this approach is insufficient to achieve that, and further investigation is needed.

3. Construction of 3D textons from 2D textons: Constructing 3D information from 2D information is a difficult task. The correspondences between the texture features and preservation of the details are important. In solid texture synthesis medial representation of the textons along with signed distance fields are used for setting these correspondences constraint during neighborhood search in synthesis. This work is an initial attempt to solve this problem. That solution is limited in the sense that the correspondences could use a different form of medial representation.

4. Computation time: Model-based texture synthesis happens in world-coordinates as opposed to the other techniques that warp cubic textures in object coordinates to fill the space. Thus, every voxel in world-coordinates has to be synthesized in this coordinate system in order to fill the space inside the model. Though this solves the problem of preserving the structural continuity in the synthesized textures, it brings computational expense. This dissertation focuses on the accuracy of the techniques more than the efficiency of them. However, improvements in the computational efficiency can be done via different techniques, such as subdivision techniques for the model.

7.3 Future Work

In the future the possibility of adding information from different sources to overcome the limitations of a specific data source will be important for various applications, such as guiding complex work processes, expressing uncertainty in the raw data, and improving communication across domain disciplines. New methods and intermediate steps must then be developed in illustrative visualizations to support the integration and visualization of different sources of information. In this section possible near-term additions to the framework are discussed first. Then the potential applications of illustrative visualization methods from this dissertation are covered. Finally, some suggestions for improving the algorithms presented in this dissertation are discussed.

7.3.1 Future directions for the framework

Some of the possible additions to the framework can be as follows:

1. Evaluation of renderings: In my dissertation evaluating the success of the illustrations generated using the MGTS framework is a hard task as for other illustrations. In general, the evaluation of renderings can be done by measuring some quality or by asking users expert on the application. Measuring the quality of the illustrative renderings is hard to achieve since it is hard to find the right quality to measure.

2. Using image-based guidance information for understanding acquired medical data sets: In this dissertation the model-based guidance information is used for constraining texture synthesis inside a model. Though this guidance information is based on the medial representation of a segmented model, which is obtained via analyzing the underlying intensity information in the acquired data set, it does not give any information about the tissues in different regions of the anatomical model. In the future, tissue-specific guidance information can be derived directly from image intensities in the acquired data set (CT or MRI) or images derived

from them. This information can be used as guidance information or for selecting region-specific textures in texture synthesis. In this way, I believe we can provide more information about the patient in an effective and understandable way to the doctors.

Using such guidance information for other datasets, especially ones that include both healthy and diseased anatomies, can be an important next step to use the approaches presented in this dissertation. One example of such an image-guided texture synthesis approach is presented in Patel et al. (2009) for the visualization of seismic data. In that approach information available in the raw seismic data is interpreted to guide the creation of geoscientific illustrations via textures. In these illustrations, textures and their features (e.g., orientation, scale, intensity) on planar surfaces are used to emphasize layers and faults in seismic data as opposed to labeling them.

3. Adapting different rendering styles: The properties of a texture, such as its scale, orientation, color, and pattern, are a common means to encode different sources of information. In the MGTS framework, the combination of these properties is used to visualize the material characteristics of the anatomical structures. Other visualization techniques could be integrated into the MGTS framework for other visualization purposes. For example, silhouette and feature lines can be added to the illustrations to emphasize the shape of objects, and hatchings can be employed to highlight the curvature of objects as in Saito & Takahashi (1990). Thus, combination and selection of different rendering styles to visualize different objects should be investigated in the future.

4. Labeling of anatomical models: Labels in visualizations give textual explanations about the data. Label placement is an important problem in visualization since labels should be placed near to the relevant information, and they should not

occlude other labels. In Luboschik et al. (2008) an approach that considers other visual elements and labels for solving label placement problem is presented for information visualization purposes. In anatomical illustrations, there are two types of labeling: internal and external. For internal labeling, medial-axis alignment is used as a common method in putting labels on the objects. In the future, this can be added to our framework. Axis-aligned labels can be put on textured models in order to give more explanations about the organs. This is an easy addition since medial representations of the models are assumed to be given in this dissertation.

5. Standard medical texture database: As indicated in item 2, in geoscientific illustrations textures are used to encode the information available in the data. To ease communication, geologists and geoillustrators use a standardized texture language for representing rock types in these illustrations. There is a need for a similar database for medical textures in order to provide a common language in doctor-doctor and doctor-patient communications. Especially in the future when image-based guidance information is used to interpret the acquired data, this will be a more important issue since encoding should use a common language to avoid possible confusions.

6. Visualization of sub-resolution detail using textures: The 2D exemplar textures used in this dissertation are assumed to have simple details. Though this is helpful for identification purposes, it should be improved in the future for clinical applications. For some applications, it can be useful to generate the missing details via different textures depending on the sub-resolution desired by the user. Microscopic textures might be utilized to obtain the desired zooming effect. Investigating methods for obtaining this effect can be an interesting future direction.

7. Visualizing uncertainty via textures: Textures synthesized using MGTS can change progressively from one region to another to visualize the material transition. This

transition effect can be used for other applications. For instance, textures can be used to show detailed pictures of something we are very uncertain about. For segmentation they can be used to allow visualizing the uncertainty of the segmentations in regions of the 3D segmented objects. The intensity values can be manipulated to express the level of uncertainty.

8. Applications besides medicine: The methodology presented in this dissertation can be applied to other applications for obtaining illustrative visualizations in order to encode information in the underlying data and identifying models inside the volume. One example can be flow visualization.

7.3.2 Future directions for methodologies

To solve the limitations of the framework and the methods in it, further investigation is necessary. In this section, possible future directions are presented to solve some of these limitations:

1. **Possible improvements for texture metamorphosis**:

   (a) Usage of different distance metrics: Integrating alternative features and landmarks into the energy equation in order to better control the transformation of the textures can be a good future direction.

   (b) Metamorphosis between multiple textures: In this dissertation interpolation between two input textures is investigated. It would be interesting to design a new optimization framework with additional penalty terms and constraints to perform the interpolation among a number of input textures.

   (c) Metamorphosis between solid textures: The aim of metamorphosis was to obtain interpolated 2D exemplar textures in this dissertation. In the future, metamorphosis can be extended to morphing solid textures.

2. **Possible directions for guidance information**:

   (a) Guidance information for multi-figure objects: Some anatomical organs are represented by multi-figure medial representations. For these objects, computation of the along-, across-, through- object parameterization should be handled in the future for computing the model-based guidance information specific to these organs.

   (b) Guidance information from s-reps: In slabular m-reps, not counting crest spokes, each atom is composed of two spokes of equal length, one pointing towards the top boundary and the other towards the bottom boundary. In such cases, both of the spokes in an atom change together when the object deforms. In s-reps, which are presented in detail in Pizer et al. (2011), the spokes can change independently. Thus, when a specific boundary feature moves on the object, its corresponding spoke moves on the medial sheet, preserving the correspondence. This correspondence preservation can be useful for obtaining consistent visualizations of similar objects.

   (c) Different-sources of guidance information: In this dissertation, one input to the framework was the segmented anatomical organs along with their medial representations. This medial representation was used for computing the guidance information. However, having a medial representation for a segmented anatomical organ may not be possible for every case. In the future, alternative ways of computing around/along/through coordinates can be investigated. One possible approach can be having model-based guidance information for an average model of a specific anatomical organ, and modifying it for different instances using the deformation between the average model and that specific instance. The guidance information on that average model could be created using a sketch-based approach.

3. **Possible directions for texture synthesis**:

   (a) Material-specific histogram-matching: The decision about how the textons are distributed in the synthesized texture is made by considering illustrations in textbooks. In the future, photographs of the organs can be used for measuring the scattering properties of a given set of base materials. Based on this guidance, different histogram-matching techniques, besides textons, can be used for texture synthesis.

   (b) Advanced parameterization methods for texture features: There are several kinds of texture features that can be used for preserving structural features in texture synthesis that were not discussed in this dissertation. For instance, I utilized an approximate medial representation for creating 3D texton structures from 2D structures. Though in this dissertation it is shown that this improved the quality of the results, in the future more precise parameterization, such as shock graphs (Siddiqi et al. (1998)), can be used for solving this problem.

   (c) Computation time: One of the limitations of the texture synthesis approach that I presented is that the computation time increases with the number of voxels inside the model. This might not be a problem in smaller objects with low detail, but it could become prohibitive with large objects with many details. Therefore, in the future adaptive strategies can be used for increasing the efficiency of the approach.

7.4 Implementation

The MGTS framework consists of three main components. All of the methods in these components have been implemented:

1. Texture metamorphosis: This component is implemented in Matlab. This component gets two 2D images as input, and it generates 2D images between them. The parameters for this component, such as the number of intermediate images, can be set by the user.

2. Model-based guidance information: This component is implemented in Visual C++. It gets the 3D mesh of the object and its medial representation as inputs, and it generates different model-based guidance information for each voxel inside the model depending on the application.

3. Texture synthesis: This component is implemented in Visual C++. This component gets 2D exemplar textures, the 3D mesh of the model, and the model-based guidance information as input. It synthesizes textures for that specific model using these inputs. For the solid textures it saves the output in the .vol file format, which can be used as input in the MGR framework. For the surface textures, it saves them as 2D output textures for each patch, which can also be read by other rendering frameworks.

This code can be found in the web page www.cs.unc.edu/~ilknurk/MGTS.

Bibliography

Agarwala, A. (1999). *Volumetric Surface Sculpting*. Masters thesis, MIT. 55, 56

Ahuja, N. & Todorovic, S. (2007). Extracting texels in 2.1d natural textures. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on* (pp. 1 –8). 121

Ashikhmin, M. (2001). Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, I3D '01 (pp. 217–226). New York, NY, USA: ACM. 38, 41, 42

Beg, M., Miller, M., Trouvé, A., & Younes, L. (2005). Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International Journal of Computer Vision*, 61(2), 139–157. 65, 66, 67, 73, 74

Blum, H. (1967). A Transformation for Extracting New Descriptors of Shape. In W. Wathen-Dunn (Ed.), *Models for the Perception of Speech and Visual Form* (pp. 362–380). Cambridge: MIT Press. 59

Bourke, P. (1996). Parametric equation of a sphere and texture mapping. Website. `http://paulbourke.net/texture_colour/texturemap/`. 26

Bovik, A., Clark, M., & Geisler, W. (1990). Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1), 55–73. 21

Bruckner, S. (2006). Interactive illustrative volume visualization techniques for exploration and communication. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06 New York, NY, USA: ACM. 7, 9, 10

Bruckner, S., Rautek, P., Viola, I., Roberts, M., Sousa, M. C., & Gröller, M. E. (2010). Hybrid visibility compositing and masking for illustrative rendering. *Computers & Graphics*, (34), 361–369. 10

Burns, M., Haidacher, M., Wein, W., Viola, I., & Gröller, E. (2007). Feature emphasis and contextual cutaways for multimodal medical visualization. In *Proceedings of Eurographics / IEEE VGTC Symposium on Visualization (EuroVis 2007)* (pp. 275–282). 10

Chaney, E., Pizer, S., Joshi, S., Broadhurst, R., Fletcher, T., Gash, G., Han, Q., Jeong, J., Lu, C., Merck, D., Stough, J., Tracton, G., Bechtel, M., Rosenman, J., Chi, Y., & Muller, K. (2004). Automatic male pelvis segmentation from ct images via statistically trained multi-object deformable m-rep models. In *American Society for Therapeutic Radiology and Oncology (ASTRO)*. 59

Chaudhuri, B. & Sarkar, N. (1995). Texture segmentation using fractal dimension. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1), 72 –77. 21

Chen, J. & Wang, B. (2010). High quality solid texture synthesis using position and index histogram matching. *The Visual Computer*, 26, 253–262. 125, 126, 128

Chen, W., Yan, Z., Zhang, S., Crow, J. A., Ebert, D. S., McLaughlin, R. M., Mullins, K. B., Cooper, R., Ding, Z., & Liao, J. (2009). Volume illustration of muscle from diffusion tensor images. *IEEE Transactions on Visualization and Computer Graphics*, 15(6), 1425–1432. xix, 52, 136, 137, 138, 149

Cross, G. & Jain, A. (1983). Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(1), 25–39. 21

Cross, G. R. (1980). *Markov random field texture models*. PhD thesis, East Lansing, MI, USA. AAI8112063. 124

Cutler, B., Dorsey, J., & McMillan, L. (2004). Simplification and improvement of tetrahedral models for simulation. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '04 (pp. 93–102). New York, NY, USA: ACM. 56

Cutler, B., Dorsey, J., McMillan, L., Müller, M., & Jagnow, R. (2002). A procedural approach to authoring solid models. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02 (pp. 302–311). New York, NY, USA: ACM. 49, 56

Dana, K. & Nayar, S. (1998). Histogram model for 3d textures. In *Computer Vision and Pattern Recognition Proceedings* (pp. 618 –624). 20

Design, D. (2001). Dosch textures: Medical visualization v3. Website. `http://www.doschdesign.com/products/textures/Medical_Visualization_V3.html`. 22

Dong, F. & Clapworthy, G. J. (2005). Volumetric texture synthesis for non-photorealistic volume rendering of medical data. *The Visual Computer*, 21(7), 463–473. 51

Dorsey, J., Edelman, A., Jensen, H. W., Legakis, J., & Pedersen, H. K. (1999). Modeling and rendering of weathered stone. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99 (pp. 225–234). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. 55, 56

Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., & Worley, S. (2002). *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., third edition. 34

Efros, A. & Leung, T. (1999). Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2 (pp. 1033–1038). 36

Efros, A. A. & Freeman, W. T. (2001). Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01 (pp. 341–346). New York, NY, USA: ACM. 36, 39, 42

Funkhouserl, T. (2002). Overview of 3d object representations. Website. `http://www.cs.princeton.edu/courses/archive/fall02/cs526/lectures/reps.pdf`. 52

Garcin, L. & Younes, L. (2005). Geodesic image matching: A wavelet based energy minimization scheme. *Lecture notes in computer science*, 3757, 349. 33, 73

Gorla, G., Interrante, V., & Sapiro, G. (2003). Texture synthesis for 3d shape representation. *IEEE Transactions on Visualization and Computer Graphics*, 9(4), 512–524. xvii, xviii, 42, 43, 45, 106, 108, 109, 110

Grigorescu, S., Petkov, N., & Kruizinga, P. (2002). Comparison of texture features based on gabor filters. *IEEE Transactions on Image Processing*, 11(10), 1160 – 1167. 21

Han, J., Zhou, K., Wei, L.-Y., Gong, M., Bao, H., Zhang, X., & Guo, B. (2006). Fast example-based surface texture synthesis via discrete optimization. *The Visual Computer*, 22, 918–925. 125

Han, Q. (2008). *Proper Shape Representation of Single Figure and Multi-Figure Anatomical Objects*. Ph.D. Dissertation, UNC Chapel Hill. 59, 62

Haralick, R. (1979). Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5), 786 – 804. 20

Hart, G., Zach, C., & Niethammer, M. (2009). An optimal control approach for deformable registration. In *IEEE Computer Society Workshop on Mathematical Methods in Biomedical Image Analysis* (pp. 9 –16). 66, 67, 69, 72, 73, 74, 81, 87

Heeger, D. J. & Bergen, J. R. (1995). Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95 (pp. 229–238). New York, NY, USA: ACM. 46

Hermosillo, G., Chefd'Hotel, C., & Faugeras, O. (2002). Variational methods for multimodal image matching. *International Journal of Computer Vision*, 50(3), 329–343. 73

Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., & Salesin, D. H. (2001). Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01 (pp. 327–340). New York, NY, USA: ACM. xiv, 42

Holm, D. (2009). Euler's fluid equations: Optimal control vs optimization. *Physics Letters A*, 373(47), 4354–4359. 33, 65

Jagnow, R., Dorsey, J., & Rushmeier, H. (2004). Stereological techniques for solid textures. *ACM Transactions on Graphics*, 23(3), 329–335. 46, 47

Julesz, B. (1981). Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802), 91–97. 21

Kabul, I., Merck, D., Rosenman, J., & Pizer, S. (2010). Model-based solid texture synthesis for anatomic volume illustration. In *Proceedings of Eurographics Workshop on Visual Computing for Biology and Medicine (VCBM)* (pp. 133–140). 117

Kabul, I., Pizer, S., Rosenman, J., & Niethammer, M. (2011). An optimal control approach for texture metamorphosis. In *Computer Graphics Forum* (pp. 2341– 2353). 64

Kopf, J., Fu, C.-W., Cohen-Or, D., Deussen, O., Lischinski, D., & Wong, T.-T. (2007). Solid texture synthesis from 2d exemplars. *ACM Transactions on Graphics*, 26(3), 2:1–2:9. xix, xx, 40, 47, 107, 117, 125, 126, 127, 128, 134, 138, 144, 145

Kwatra, V., Essa, I., Bobick, A., & Kwatra, N. (2005). Texture optimization for example-based synthesis. *ACM Transactions on Graphics, SIGGRAPH 2005*, 24(3), 795–802. 36, 40, 109

Kwatra, V., Schodl, A., Essa, I., Turk, G., & Bobick, A. (2003). Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003*, 22(3), 277–286. 36, 39

Levoy, M. (1988). Display of surfaces from volume data. *Computer Graphics and Applications, IEEE*, 8(3), 29 –37. 10

Levy, B., Petitjean, S., Ray, N., & Maillo t, J. (2002). Least squares conformal maps for automatic texture atlas generation. In ACM (Ed.), *ACM SIGGRAPH conference proceedings* (pp. 362–371). 27

Li, W., Han, Y.-y., & Chen, J.-x. (2008). Triangular-patch based texture synthesis over arbitrary surfaces. In *Image and Signal Processing, 2008. CISP '08. Congress on*, volume 4 (pp. 441 –445). 45

Liang, L., Liu, C., Xu, Y.-Q., Guo, B., & Shum, H.-Y. (2001). Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 20(3), 127–150. 39

Liu, Y., Lin, W.-C., & Hays, J. (2004). Near-regular texture analysis and manipulation. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04 (pp. 368–376). New York, NY, USA: ACM. xiii, 17, 32

Liu, Z., Liu, C., Shum, H.-Y., & Yu, Y. (2002). Pattern-based texture metamorphosis. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications* (pp. 184–191). Washington, DC, USA: IEEE Computer Society. 31

Lu, A. & Ebert, D. S. (2005). Example-based volume illustrations. In *Proceedings of IEEE Visualization* (pp. 655–662). 52, 151

Lu, S. & Fu, K. (1979). Stochastic tree grammar inference for texture synthesis and discrimination. *Computer Graphics and Image Processing*, 9(3), 234 – 245. 21

Luboschik, M., Schumann, H., & Cords, H. (2008). Particle-based labeling: Fast point-feature labeling without obscuring other visual features. *IEEE Transactions on Visualization and Computer Graphics*, 14(6), 1237 –1244. 163

Manian, V. B. & Vásquez, R. E. (2003). Texture analysis and synthesis: a review of recent advances. In Z.-u. Rahman, R. A. Schowengerdt, & S. E. Reichenbach (Ed.), *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 5108 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference* (pp. 242–250). 17

Materka, A., Strzelecki, M., Analysis, T., Review, M. A., Materka, A., & Strzelecki, M. (1998). *Texture analysis methods - a review*. Technical report, Institute of Electronics, Technical University of Lodz. 17

Matusik, W., Zwicker, M., & Durand, F. (2005). Texture design using a simplicial complex of morphable textures. *ACM Transactions on Graphics*, 24(3), 787–794. 32

Merck, D. (2009). *Model-Guided Rendering for Medical Images*. Ph.D. Dissertation, UNC Chapel Hill. xviii, 7, 8, 28, 93, 95, 96, 107, 114, 115, 116, 147, 152, 154

Miller, M. I., Trouve, A., & Younes, L. (2006). Geodesic shooting for computational anatomy. *Journal of Mathematical Imaging and Vision*, 24, 209–228. 72

Miller, M. I. & Younes, L. (2001). Group actions, homeomorphisms, and matching: A general framework. *International Journal of Computer Vision*, 41(1/2), 61–84. 33, 73

Mirmehdi, M., Xie, X., & Suri, J. (2008). *Handbook of Texture Analysis*. World Scientific. 19

Nealen, A. & Alexa, M. (2003). Hybrid texture synthesis. In *Proceedings of the 14th Eurographics workshop on Rendering*, EGRW '03 (pp. 97–105). Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. 36

Netter, F. H. (2009). *Atlas of Human Anatomy*. Rittenhouse Book Distributors Inc. xvii, 2, 5, 9, 22, 97, 118, 140

Ojala, T., Inen, M. P., Member, S., & A, T. M. (2002). Multiresolution grayscale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 20

Owada, S., Harada, T., Holzer, P., & Igarashi, T. (2008). Volume painter: Geometry-guided volume modeling by sketching on the cross-section. In *Proc. Eurographics Symposium on Sketcy-Based Interfaces and Modeling 2008 (SBIM 08)* (pp. 9–16). 57

Owada, S., Nielsen, F., Okabe, M., & Igarashi, T. (2004). Volumetric illustration: designing 3d models with internal textures. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (pp. 322–328). New York, NY, USA: ACM. 49, 51

Paget, R. & Longstaff, I. D. (1998). Texture synthesis via a noncausal nonparametric multiscale markov random field. *IEEE Transactions on Image Processing*, 7, 925–931. 35

Patel, D., yvind Sture, Hauser, H., Giertsen, C., & Grller, M. E. (2009). Knowledge-assisted visualization of seismic data. *Computers and Graphics*, 33(5), 585 – 596. 8, 23, 162

Perlin, K. (2002). Improving noise. *ACM Transactions on Graphics*, 21, 681–682. 34

Pietroni, N., Cignoni, P., Otaduy, M. A., & Scopigno, R. (2010). Solid-texture synthesis: A survey. *IEEE Computer Graphics and Applications*, 30(4), 74–89. 45, 51

Pietroni, N., Otaduy, M. A., Bickel, B., Ganovelli, F., & Gross, M. (2007). Texturing internal surfaces from a few cross sections. *Computer Graphics Forum*, 26(3), 637–644. 49, 51

Pizer, S., Fletcher, P., Joshi, S., Gash, A., Stough, J., Thall, A., Tracton, G., & Chaney, E. (2005). A method and software for segmentation of anatomic object ensembles by deformable m-reps. *Med Phys*, 32(5), 1335–1345. 59

Pizer, S., Fletcher, T., Fridman, Y., Fritsch, D., Gash, A., Glotzer, J., Joshi, S., Thall, A., Tracton, G., Yushkevich, P., & Chaney, E. (2003). Deformable m-reps for 3d medical image segmentation. *International Journal of Computer Vision - Special UNC-MIDAG issue*, 55(2), 85–106. 59

Pizer, S., Jung, S., Goswami, D., Zhao, X., Damon, J., Huckermann, S., & Marron, S. (2011). Nested sphere statistics of skeletal models. In *Proc. Dagstuhl Workshop on Innovations for Shape Analysis: Models and Algorithms*. 165

Praun, E., Finkelstein, A., & Hoppe, H. (2000). Lapped textures. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series (pp. 465–470). 39, 42, 43, 45, 105

Qin, X. & Yang, Y.-H. (2007). Aura 3d textures. *IEEE Transactions on Visualization and Computer Graphics*, 13(2), 379–389. 47

Ramanarayanan, G. & Bala, K. (2007). Constrained texture synthesis via energy minimization. *IEEE Transactions on Visualization and Computer Graphics*, 13(1), 167 –178. 42

Ray, N., Lvy, B., Wang, H., Turk, G., & runo Vallet, B. (2009). Material-space texturing. *Computer Graphics Forum*. xv, xvi, 32, 81, 82, 85, 86

Rosenberger, A., Cohen-Or, D., & Lischinski, D. (2009). Layered shape synthesis: automatic generation of control maps for non-stationary textures. *ACM Transactions on Graphics*, 28, 107:1–107:9. 18

Ruiters, R., Schnabel, R., & Klein, R. (2010). Patch-based texture interpolation. *Computer Graphics Forum*, 29(4), 1421–1429. xvi, 32, 86

Saito, T. & Takahashi, T. (1990). Comprehensible rendering of 3-d shapes. *SIGGRAPH Computer Graphics*, 24, 197–206. 162

Siddiqi, K. & Pizer, S. (2008). *Medial Representations: Mathematics, Algorithms and Applications*, (pp. 1–450). Springer. 59, 93

Siddiqi, K., Shokoufandeh, A., Dickenson, S., & Zucker, S. (1998). Shock graphs and shape matching. In *Computer Vision, 1998. Sixth International Conference on* (pp. 222–229). 166

Soler, C., Cani, M.-P., & Angelidis, A. (2002). Hierarchical pattern mapping. In T. Appolloni (Ed.), *29th International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2002, July, 2002* (pp. 673–680). San Antonio, Texas, Etats-Unis: ACM. 39

Takayama, K., Ashihara, T., Ijiri, T., Igarashi, T., Haraguchi, R., & Nakazawa, K. (2008a). A sketch-based interface for modeling myocardial fiber orientation that considers the layered structure of the ventricles. *The Journal of Physiological Sciences*, 58(7), 487–492. xvii, 92, 98, 99, 102

Takayama, K., Igarashi, T., Haraguchi, R., & Nakazawa, K. (2007). A sketch-based interface for modeling myocardial fiber orientation. In *SG '07: Proceedings of the 8th international symposium on Smart Graphics* (pp. 1–9). Berlin, Heidelberg: Springer-Verlag. 28

Takayama, K., Okabe, M., Ijiri, T., & Igarashi, T. (2008b). Lapped solid textures: filling a model with anisotropic textures. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (pp. 1–9). New York, NY, USA: ACM. xiv, 28, 29, 50, 56, 57, 118

Takayama, K., Sorkine, O., Nealen, A., & Igarashi, T. (2010). Volumetric modeling with diffusion surfaces. In *ACM SIGGRAPH Asia 2010 papers*, SIGGRAPH ASIA '10 (pp. 180:1–180:8). New York, NY, USA: ACM. 57

Thall, A. (2002). Fast c2 interpolating subdivision surfaces using iterative inversion of stationary subdivision rules. `http://midag.cs.unc.edu/pub/papers/ThallTR02-001.pdf`. 62

Tong, X., Zhang, J., Liu, L., Wang, X., Guo, B., & Shum, H.-Y. (2002). Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics*, 21, 665–672. 38, 125

Trouve, A. & Younes, L. (2005). Metamorphoses through lie group action. *Foundations of Computational Mathematics*, (pp. 173–198). 33, 65

Turk, G. (2001). Texture synthesis on surfaces. In *SIGGRAPH* (pp. 347–354). 42, 44

Turk, G. & O'Brien, J. F. (1999). Shape transformation using variational implicit functions. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99 (pp. 335–342). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. 99

Viola, I., Hauser, H., & Ebert, D. (2010). Editorial note for special section on illustrative visualization. *Computers & Graphics*, 34(4), 335–336. 10

Ware, C. (2004). *Information Visualization, Second Edition: Perception for Design (Interactive Technologies)*. Morgan Kaufmann. 23

Wei, L.-Y. (2003). Texture synthesis from multiple sources. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications* (pp. 1–1). New York, NY, USA: ACM. 32, 47

Wei, L.-Y. (2004). Tile-based texture mapping on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, HWWS '04 (pp. 55–63). New York, NY, USA: ACM. 27

Wei, L.-Y. & Levoy, M. (2001). Texture synthesis over arbitrary manifold surfaces. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series (pp. 355–360). 42, 43, 44

Welsh, T., Ashikhmin, M., & Mueller, K. (2002). Transferring color to greyscale images. *ACM Transactions on Graphics*, 21(3), 277–280. 144

Wolfe, R. (1997). Teaching texture mapping visually. *SIGGRAPH Computer Graphics*, 31. xiv, 24, 25, 26, 27, 28

Wu, Q. & Yu, Y. (2004). Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics*, 23(3), 364–367. 39

Ying, L., Hertzmann, A., Biermann, H., & Zorin, D. (2001). Texture and shape synthesis on surfaces. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering* (pp. 301–312). 44

Zhang, J., Zhou, K., Velho, L., Guo, B., & Shum, H.-Y. (2003). Synthesis of progressively-variant textures on arbitrary surfaces. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (pp. 295–302). New York, NY, USA: ACM. 32, 138, 139