Tutorial Organizers: Mario Costa Sousa, David S. Ebert, Ivan Viola

Presenters (In alphabetical order):

Bill Andrews *Medical College of Georgia*	bandrews@mcg.edu www.mcg.edu/medart/
Stefan Bruckner *Vienna University of Technology*	bruckner@cg.tuwien.ac.at www.cg.tuwien.ac.at/staff/StefanBruckner.html
Wei Chen *Zhejiang University*	chenwei@cad.zju.edu.cn www.cad.zju.edu.cn/home/chenwei/
Carlos D. Correa *University of California, Davis*	correac@cs.ucdavis.edu vis.cs.ucdavis.edu/~correac/.
David S. Ebert *Purdue University*	ebertd@purdue.edu www.ecn.purdue.edu/~ebertd
Mario Costa Sousa *University of Calgary*	mario@cpsc.ucalgary.ca www.cpsc.ucalgary.ca/~mario
Ivan Viola *University of Bergen*	ivan.viola@uib.no http://www.ii.uib.no/vis/team/viola/

Abstract

The area of illustrative visualization is concerned with developing methods to enhance the depiction of scientific data based on principles founded in traditional illustration. The illustration community has century-long experience in adapting their techniques to human perceptual needs in order to generate an effective depiction which conveys the desired message. Thus, their methods can provide us with important insights into visualization problems.
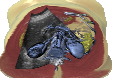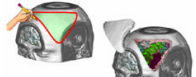
In this tutorial, the concepts in illustrative visualization are reviewed. An important aspect here is interaction: while traditional illustrations are commonly only presented as static images, computer-assisted visualization enables interactive exploration and manipulation of complex scientific data. Only by coupling illustrative visualization with effective interaction techniques its full potential can be exploited.
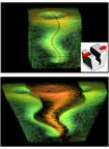
The tutorial starts with a detailed description of the entire traditional medical illustration production pipeline (techniques, tools, etc.) describing limitations and specific features to be researched and developed for more advanced tools. We then proceed discussing the importance and power of abstraction and interface issues in illustrative visualization. We present different ways of achieving abstraction in interactive settings discussing flexible representations for representing artistic visual styles. Next, we introduce the importance of intuitive interaction for illustrative visualization describing sketch-based approaches as an intuitive way of manipulating and exploring volumetric datasets. In the last part of the tutorial we present techniques for deforming volumes in various ways inspired by traditional illustration techniques such as the depiction of surgical procedures. We also describe how to deform and render in an illustrative fashion using by-example approaches.

Session/Topics	Speaker(s)
Historical Perspective on Concepts & Techniques of Traditional Medical & Scientific Illustration	Andrews
Rooted in the Renaissance and the rediscovery of Classical works on science and medicine, traditional scientific and medical illustration has evolved greatly in the centuries since. This evolution has occurred in tandem with advances in scientific and medical knowledge and understanding, as well as with advances in communications theory and technology. From allegorical to documentary and from instructive to interpretive, the conceptual approaches employed by illustrators working in these fields have changed and adapted to meet the needs of their publics, and to take advantage of technological advances. This section of the tutorial provides a brief, insightful survey of historic visualization and illustration concepts, contextual settings, and relevant media techniques.	
Interactive Illustrative Rendering with Style	Ebert, Bruckner
First part will focus on the determination of the appropriate representation of material for the user, their experience, and their task. The creation of effective visual representations needs to be based on the intended user and their intended use of the system. Description of approaches for creating not only illustrative representations that provide the appropriate level of representation and highlight the relevant material for the user's task, but also on the design of the system interface for making the system understandable to the user. The second part will focus on visual style representations for illustrative visualization. As different rendering styles are an effective means for accentuating features and directing the viewer's attention, an interactive illustrative visualization system needs to provide an easy-to-use yet powerful interface for changing these styles. The lecture will review existing approaches for stylized rendering and discuss practical considerations in the choice of an appropriate representation for visual styles. Additionally, a high-level approach for mapping volumetric attributes to different illustrative styles will be discussed.	

Intuitive and Ergonomic Interaction in Illustrative Visualization	Viola, Sousa

Illustrative visualization primarily deals with easy-to-understand display of complex underlying data, however, to enable exploration and direct *contact* with the data, easy-to-handle interaction is equally important as the visual part. Ergonomic interaction is demonstrated in user-centric mechanisms for data presentation such as guided navigation through classified volumetric data or story-telling for volume visualization. Besides approaches aiming at presentation, we describe how interactive illustrative visualization is being embedded into novel medical intervention procedures.

We will also review the state-of-the-art of sketch-based interfaces and modeling (SBIM) for scientific visualization, including different aspects and inspiration factors brought from traditional medical/scientific illustration principles, methods and practices We will describe unique techniques and problems, including presentation of systems, algorithms and implementation techniques focusing on interactive SBIM for illustrative volume graphics.

Example-based illustrative Rendering and Deformation	Correa, Chen

First part focuses on a methodology for generating visualizations that depict deformation, in order to enhance the view of hidden features or to depict a complex procedure, such as a surgical operation. To obtain high-quality images, a number of considerations need to be taken into account, such as sampling, lighting and composition of volumes undergoing deformation. The lecture also presents an architectural view of the system and algorithmic details for its implementation using contemporary graphical processing units (GPUs). In addition, selective deformation can be obtained with the use of user-defined masks and segmentation information. This lecture also describes how to incorporate these aspects into the illustrative deformation pipeline.

Second part will focus on how to learn illustration styles from traditional illustrations or measured datasets, including the color, texture, structure and shape styles. We will describe how to modify and decorate a 2D illustration by simulating the shape styles of another 2D example using differential based mesh manipulation techniques. Our second scheme aims to change a 3D model template with shape styles from 2D examples. In addition, we will present an efficient shape-aware technique to abstract the boundaries of 3D models for achieving smooth boundary effects. Finally, we will introduce how to learn and re-use other styles such as colors, textures under an example-based volume illustration framework. The driving techniques for these goals are constrained texture synthesis and differential coordinates based shape manipulation.

Image credits

	© 1999 - 2006 Fairman Studios, LLC. All Rights Reserved.
	© Nikolai Svakhine, Yun Jang, David S. Ebert and Kelly Gaither, "Illustration and Photography Inspired Visualization of Flows and Volumes", Proceedings of IEEE Visualization Conference 2005, Minneapolis, October 23 - 25, 2005
	© Copyright M. Burns et al. *Feature Emphasis and Contextual Cutaways for Multimodal Medical Visualization*. In Proc. of EUROVIS 2007. Used by Permission
	© Copyright *VolumeStudio*, University of Calgary, 2007, Used by Permission

	© *Modeling Plant Structures Using Concept Sketches* Anastacio,F., Sousa, M.C., Samavati, F., Jorge, J. 4th International Symposium on Non Photorealistic Animation and Rendering (NPAR '06)
	© S. Bruckner, M. E. Gröller. Exploded Views for Volume Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077-1084, 2006.
	© Illustrative Deformation for Data Exploration. Carlos Correa, Deborah Silver and Min Chen. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2007),* vol. 13, no. 6, Nov.-Dec. 2007.
	© *Shape-aware Volume Illustration*. Wei Chen, Aidong Lu, David S.Ebert. Computer Graphics Forum (Proceedings of Eurographics 2007).
	© Bill Andrews. All Rights Reserved.

Historical Perspective on Concepts & Techniques of Traditional Medical & Scientific Illustration

Bill Andrews, *Medical College of Georgia*

bandrews@mcg.edu
www.mcg.edu/medart/

Just as fine art is a reflection of the culture and times in which it was created, images of science and medicine reflect the contextual environment of their creators. This presentation provides an insightful survey of historic visualization and illustration concepts, contextual settings, and relevant media techniques.

Pre-Renaissance Era

The Middle ages or Medieval period extends from the fall of the Roman Empire until the advent of the Renaissance, from roughly 400 AD to 1350 AD. In the later half of this period is marked by the rediscovery and interpretation of the knowledge and culture of Classical Greece and Rome. Philosophically, there is the ascension of Christianity and the expansion of Catholic doctrine, with an all-pervading faith in the divine order of heaven and earth through time without end. In contrast, the rediscovered scientific knowledge and technology encourage the birth of reason and empiricism. Infusions of technology and new ideas such as algebra, from the Middle East and Asia caused perturbations in the status quo and created new opportunities. Late in this period, arrival of the bubonic plague from the east shook the "natural order" of European civilization. While the Middle Ages are often portrayed as a period of dogmatic religious dominance and technological stagnation, this is not so. The stage was being set—nearly all the art, architecture, technology and philosophy that we celebrate about the Renaissance had its roots in the Middle Ages.

In the beginning of this Pre-Renaissance era, communication technology was dominated by monastic scribes working on vellum, creating one document at a time. In the 1100's, effective and efficient paper-making technology arrived in Europe. It arrived almost contemporaneously with wood block printing. With these two innovations, knowledge became an affordable commodity. The church no longer had a monopoly on mass communication. However, preparation of the wood blocks was laborious (and unforgiving), and the blocks have a limited lifespan measured in 100s of copies. The grain of the wood and fineness of the carving instrument limited image resolution and could introduce artifacts to the image.

Quintessential images from science and medicine from this period include those in the Margarita Philosophica by Gregor Reisch (1467-1525), a Carthusian monk from Freiburg, and the Fasciculus Medicinae by Johannes de Ketham, a German physician. While both these works were technically printed during the Renaissance, they feature images that are very much Pre-Renaissance. The dissected male figure from the Margarita (http://www.kcl.ac.uk/depsta/iss/library/speccoll/bomarch/bomapril06.html) dogmatically depicts Galenic anatomy. Galen of Pergamum was a second century Greek physician. Unfortunately, when his writings were rediscovered during the Middle Ages they were accepted as gospel— including the incorrect anatomy. *The Anatomy Lesson* from the Fasciculus (http://www.nlm.nih.gov/exhibition/historicalanatomies/ketham_home.html) shows a typical early anatomical dissection—the physician reading Galen while a barber-surgeon does the actual cutting. This image captures the conflict between accepted knowledge (Galen's writings) and empirical observation.

The Renaissance

For the purposes of this talk, the Renaissance begins in the middle 1300s and extends to the early 1600s. Italy is the epicenter of this Renaissance. Described as the rebirth of Classical Greek and Roman culture, it is built on the foundations set during the late Middle Ages. Empiricism replaces dogma, reason gains ascendancy over faith, and secular powers vie with the church-dominated status quo. Original knowledge was pursued via the study of hidden similarities and relationships between physical objects, as well as between physical objects and metaphysical beings (via allegory, parable and symbolism). During this period, those with the means began collecting items from the natural world and from other cultures, giving rise to curiosity cabinets (kunstkammers).

In communications technology, the German printer Johannes Gensfleisch zur Laden zum Gutenberg (c. 1400 – 1468) took the next great leap, with movable metal type (http://www.mainz.de/Gutenberg/museum.htm). His first printing project was a Bible (1439). Perhaps presaging the standard business model for most new communication technologies, he went bankrupt. Printing with engraved metal plates was also invented at this time, though its use would remain rare for a

very long time. Another new technology emerging from the Renaissance was oil paint. Advances in pigment chemistry become increasingly important for communications from this time forward.

Quintessential science and medical imagery include <u>De Humani Corporis Fabrica</u> (1543) by Andreas Vesalius of Brussels (1514 – 1564) (http://vesalius.northwestern.edu/). The text is printed with movable type, with images printed from wood blocks. This book revolutionized anatomical and medical education. Vesalius did his own dissections; the illustrations were based on observation; the text and illustrations are integrated into a coherent whole; and it was mass produced (several hundred copies are still in existence). The frontispiece from the <u>Fabrica</u> provides an excellent counterpoint to *The Anatomy Lesson* from the <u>Fasciclus</u>.

Other representative imagery includes <u>Vier Bücher von Menschlicher Proportion</u> (1528) by the German artist Albrecht Dürer (1471 – 1528). This posthumously published book depicts Dürer's research into the morphometric proportions of the human form (http://www.nlm.nih.gov/exhibition/historicalanatomies/durer_home.html). As with Vesalius' <u>Fabrica</u>, the <u>Proportion</u> was printed with a combination of movable metal type and wood block figures, but it also included some engraved metal plates.

The Baroque

The Baroque, sometimes called the Age of Reason, is an extension of the Renaissance and extends roughly from 1600 to 1700. In the sciences, it is noteworthy for the rise of ordering and classification as a source of discoveries (distinction vs. similitude). There is also a growing scientific rift between the focus on objective analysis by the rationalists and the subjective experience of the empiricists. Building on the concept of curiosity cabinets inaugurated in the Renaissance, there is a general proliferation of protomuseums, botanical gardens and menageries. Artistically, Baroque artists stressed emotion, variety, ornamentation and movement.

During this period logarithms, calculus and the slide rule are created; however, it would be a few years before their influence in scientific and medical imagery would be noticeable.

Precision instrumentation begins to make a real impact in science and medicine. Antony van Leeuwenhoek (1632 – 1723), the father of microscopy, was a lens grinder in Delft (http://www.ucmp.berkeley.edu/history/leeuwenhoek.html). Galileo Galilei (1564 – 1642), the father of modern science, envisioned a new cosmology through his telescope (http://www.galilean-library.org/). This period marks the advent of machine-aided vision.

In communications, hi-tech meant engraving on copper plates. The use of metal printing plates meant that 1,000s of copies featuring images with very fine detail could be printed.

Quintessential images of science and medicine from the Baroque can be seen in the atlas <u>Ontleding des Menschelyken Lichaams</u> by the anatomist Govard Bidloo (1649-1713) from Amsterdam (http://www.nlm.nih.gov/exhibition/historicalanatomies/bidloo_home.html). The *Allegorical Titlepage* sums up the Baroque style nicely and is also a fine example of copperplate engraving.

The Age of Enlightenment

This period begins around 1650 and extends through the 1700s. It is an age of optimism, powered by an intellectual movement which advocated reason as the basis for authority. The movement sought to improve the human condition after centuries of unquestioned tradition, superstition and tyranny. The absolute authority of the nobility and the church declined. Logic and science became ascendant.

Newton, Leibniz, Hooke, Lavoisier and others explored the principals and methods that laid the foundations for the coming Industrial Revolution. Experimentation begins to overtake pure empiricism. New heights of precision instrumentation are reached, propelling a desire to explore, measure and document everything.

Bernhard Siegfried Albinus (1697-1770), a physician from Frankfurt, and his artist Jan Wandelaar (1690-1759), from Leiden went to great lengths to create the "perfect" anatomy text They began with the search for the ideal specimen. Albinus posed the cadaver after successive layers of dissection. Then Wandelaar drew the cadaver using an anti-parallax contraption. They compared this work against numerous ancillary cadaver dissections to arrive at a final normative ideal. It is still a work of unparalleled accuracy and beauty. (http://www.nlm.nih.gov/exhibition/historicalanatomies/albinus_home.html).

In medicine there is a drive to define normal and to study the abnormal. Surgery, as a separate profession, emerges. The work of the Hunter brothers, William and John, of Scotland provide fine examples. Anatomist and Royal physician William Hunter (1718-1783) created the Anatomia Uteri Humani Gravidi Tabulis Illustrata, considered one of the most beautiful medical texts of all time (http://www.nlm.nih.gov/exhibition/historicalanatomies/hunterw_home.html). One of the reasons that the book is remarkable is that the dissected figures are reproduced at life size. John Hunter (1728 – 1793) received no formal education in medicine, but went on to be one of the greatest surgeons, anatomists and naturalists of his age. He also did experimental surgery, including transplantation (http://www.rcseng.ac.uk/museums/history/johnhunter.html).

New printing methods are created—the mezzotint process and the aquatint process. Both allow the printing of a full tonal range of shades, from black to white.

The Industrial Age

The Industrial Age extends from roughly 1750 to 1900, though some would argue that it has not yet ended. This age is marked by the triumph of mechanization and industry over nature and manpower. The age of scientific and medical specialization begins. In medicine, the study of diseases and abnormalities comes into vogue.

The publication of technical journals and periodicals by philosophical and scientific societies helped to spread technical innovations. The invention of lithography, and later the powered printing press, facilitated the dissemination of all this new information, as did the advent of the steam-powered railway. In the early 1800s, advances in chemistry led to photography. The age of mechanochemical vision had dawned. By the later half of the 1800s, telegraphy makes distributed network communications possible over long distance.

During the American Civil War, Mathew Brady captured images of field surgeries (http://www.sonofthesouth.net/leefoundation/amputation.htm). Neurologist Guillaume Benjamin Amand Duchenne (1806-1875) from Boulogne sur Mer, France, produced two major treatises entitled De l'Électrisation localisée (1855) and Physiologie des Mouvements (1867) (http://www.whonamedit.com/doctor.cfm/950.html). His methods were a bit unorthodox by today's standards—he applied electric currents to the muscles causing them to contract, and then photographed them. Between 1831 and 1854 physician Jean Baptiste Marc Bourgery (1797–1849) and artist illustrator Nicolas Henri Jacob (1782–1871) published Traité Complet de l'Anatomie de l'Homme Comprenant la Medecine Operatoire in eight volumes. This remarkable book includes color printing using the lithograph method.

The Modern Era

Also known as the Second Industrial Age, the Modern Ear extends from about 1850 to the end of World War II. This era is marked by the ascendancy of chemistry, assembly line manufacturing, electric power, the automobile, the computer, telecommunications and powered flight.

The Impressionists capitalize on advances in chemistry. In photography, the halftone screen process enables tonal printing. Photomicrography comes into being, capturing the very small. In the early 1900s, the safety bulb makes flash photography practical and safe. In printing, the rotogravure press is invented, allowing rapid color printing. Though still expensive to produce, more and more books are printed in color. In the early 1900s, wireless broadcast radio communications are a commercial success, and television is under research.

The advent of safe and effective anesthetic agents opens new frontiers in surgery. Medical artists no longer document anatomy, they now describe novel operative techniques.

The Post-Modern Era

Beginning at the close of World War II, this era ends with Disco in 1970s. The Post-Modern Ear is noted by the advent of nuclear power and advances in particle physics, rocketry, television, home appliances, power tools, plastics, antibiotics, the pill, and extrasensory visualization. With foundations established by disillusionment arising from horrors of World War II, postmodernism is a cultural movement lacking a clear organizing principle. It embodies complexity, contradiction, ambiguity, diversity, and interconnectedness.

Advances in stroboscopic photography, extended spectrum films, electron microscopy and so on open up

the field of scientific visualization beyond the limits of human vision. The stunning images by Lennart Nilsson are exemplary (http://www.lennartnilsson.com/human_body.html). Broadcast television makes it possible to share moving pictures with an entire population simultaneously.

In medicine, there is a new focus at the cellular function, physiological processes and biochemistry. The work by physician-artist Frank H. Netter typifies this focus (http://www.netterimages.com/artist/netter.htm). Netter was known for his interpretive approach to anatomy and physiology, as opposed to a documentary approach, in order to serve his didactic purpose.

The Computer Age

For the purposes of this presentation, the computer age begins with the advent of the desk-top computer in the 1970s. It continues today. I'm no expert on computers or their impact on society, but I am amused by them and occasionally find them useful. In my career, I've gone from the "fun" of creating daisy-wheel images with the letters X and O on fan-fold paper to being able to model complex proteins in 3D and change them over time. Thirty years ago, medical illustrators were low-tech, low-cost image specialists—all we needed was paper, pen and ink. Now, in order to create interactive 3D animations, we consume more raw computing power than most people on the academic medical campus.

The accompanying sample by XVIVO and Harvard Medical School depicts the current approach to didactic medical imagery (http://multimedia.mcb.harvard.edu/media.html). Note the influences from the entertainment industry.

The Information Age

For the purposes of this presentation, the Information Age begins with the advent of the Internet, and is ongoing. It pervades all aspects of our culture and economy. It can be characterized by an obsessive desire to collect, organize, process, distribute, massage, interpret, repurpose, and perhaps use massive amounts of data instantly, interactively, anywhere.

In medicine and science, we can now take a string of numbers and turn it into an image, a visualization of some theoretical construct. As an example, the ARGO Genome Browser from the Broad Institute and MIT will serve (http://www.broad.mit.edu/annotation/argo/).

In essence, imagery has been freed from the need to portray reality and is now limited only by our conceptual abilities and communication needs.

Tailoring Interfaces and Levels of Representation in Illustrative Visualization and Rendering
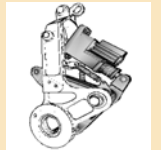
David S. Ebert
Electrical & Computer Engineering
Purdue University
ebertd@purdue.edu

Approach to Effective Visual Representations (ala Bertin)

Understand the problem or question

Determine data needed for solution and its characteristics

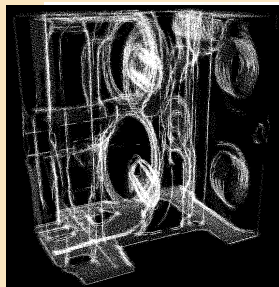*Determine effective visual representation*

- Utilize perception, design, illustration, and advanced rendering techniques
- Interactivity, accuracy, and reproducibility are vital

Illustrative Visualization: Overview

Abstract away unimportant details

Illustrative Visualization: Overview

Utilize attentive focus to emphasize data

Illustrative Visualization: Overview

Utilize illustration principles and techniques
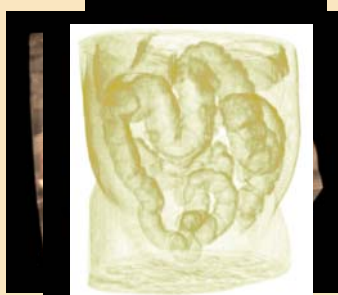
Illustrative Visualization Approach

What to show?

- Incorporate principles from technical illustration

How to show it?

- Develop a toolbox of illustrative techniques

How to implement it?

- Adapt volume rendering pipeline to volume illustration

How to create an appropriate interface

- Interactive design with user participation

Toolbox of Techniques

Feature enhancement
- Boundary enhancement
- Silhouette enhancement

Depth and orientation techniques
- Aerial perspective
- Intensity depth cueing
- Oriented fading
- Halos
- Tone shading

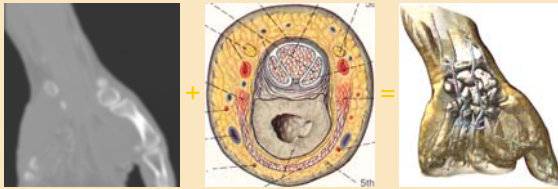---

Approaches to Interfaces

Example-based interfaces

Multi-level interfaces

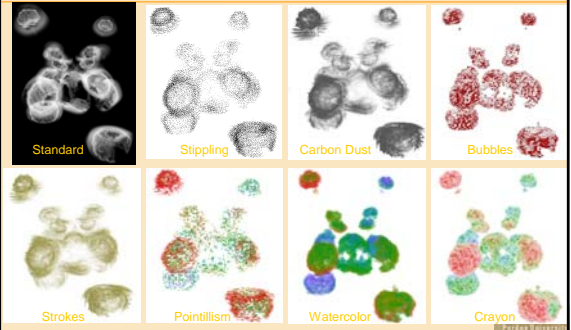*Interfaces and adaptation to task*

Interfaces and adaptation to devices
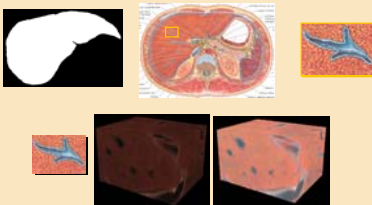

Example-BasedVolume Illustration

Multiple Styles - Iron Protein

Standard Stippling Carbon Dust Bubbles
Strokes Pointillism Watercolor Crayon

Example-based Illustrative Visualization

Simulate style of professional illustrators & simplify user interaction

---

Color Transfer

Example Source 1-1 2-2
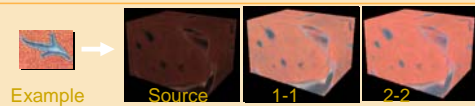
Current approaches
- "Color transfer between images", E. Reinhard et al. 2001
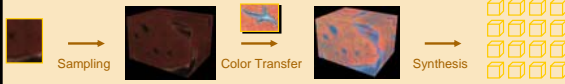- "Transferring color to greyscale images", W. Welse et al. 2002

Automatic transferring process

Two assumptions:
- Simpler representation & similar object distributions

Example-based Rendering

Sampling Color Transfer Synthesis
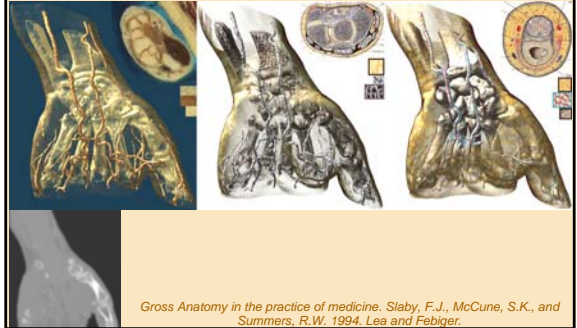
For each object – one set of textured cubes

Segmented datasets: object ID – one cube set

Un-segmented datasets: opacity – cube sets

Only two user interactions: sample & illustration examples
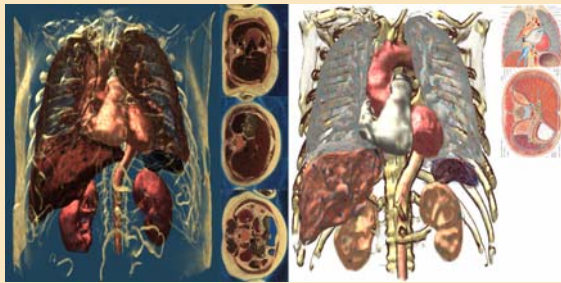
PURPL
Purdue University

Example-based Rendering - Hand

Gross Anatomy in the practice of medicine. Slaby, F.J., McCune, S.K., and Summers, R.W. 1994. Lea and Febiger.

Example-based Rendering - Abdomen

Sobotta Atlas of human anatomy. Staubesand, J. 1990. Urban and Schwarzenberg Baltimore-Munich.

Goal of Interactive Medical Illustrative Visualization Process

PURPL
Purdue University

Hierarchical Transfer Functions

Usually transfer functions control (color, opacity)

They can also be used to control parameters/contribution of the effects

Design multiple transfer functions to control:
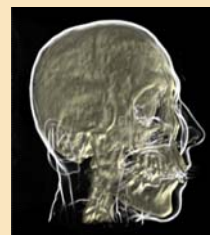
- Sketch
- Illumination
- etc.

This allows applying different combinations of effects to different materials

PURPL
Purdue University

Example

Sketch applied to 'skin' material

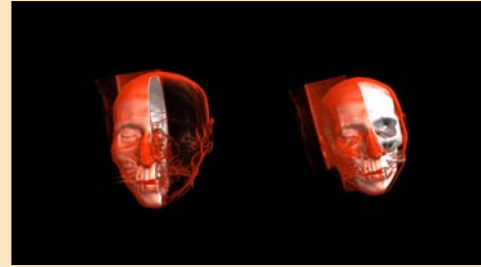*Illumination applied to 'bone' material*

PURPL
Purdue University

3

Next Step: Zones

Zones are regions of the volume space
Zones can have different applied styles
Zones can be either ellipsoids or rectangles
This allows large set of possible effects

Zones With Different Styles

Common Medical Illustration Components

Organ/tissue boundary emphasis
- Strong use of silhouetting techniques

Variation of level of representation
- Focus + context techniques

Common representation vocabulary
- Colors
- Styles

IVIS Animation

Medical Motifs

Motifs are settings upon which illustrator can quickly build styles specification

Example:
- Anatomical illustration
- Surgical simulation
- Different levels of expertise for intended users

Medical illustrations

Foot bone structure from anatomy textbook
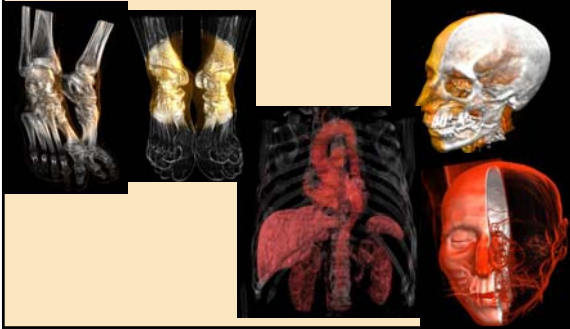
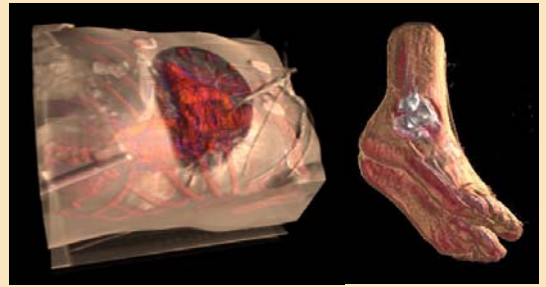Same structure shown by IVIS with Visible Human foot dataset

Anatomical Illustration

Surgical Training

Images by Nikolai Svakhine

Levels of Expertise

Novice
- Frequently overwhelmed by the quantity and complexity of data presented during training
- Must learn to develop their attentive focus and unconsciously orient the structures in the data for reference
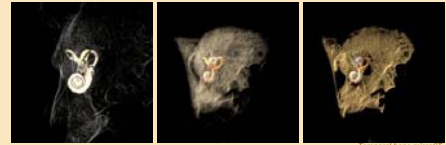
Expert
- Has necessary experience to subjugate data details that provide context
- Can quickly focus on the specific portion of the data and relevant structures

Levels of Representation

Motifs are designed with help/feedback from professional medical illustrator

Temporal bone microCT

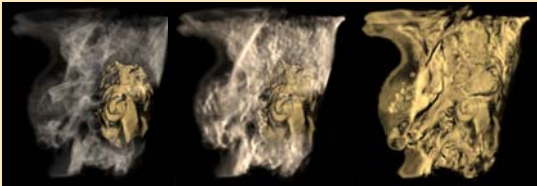Schematic/simplistic representation of the cochlea and semicircular canals

A more complex representation of the same area, with more detail on surrounding bone
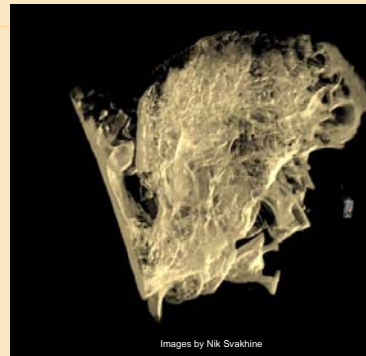
Almost 'realistic' representation

Levels of Representation

Temporal bone microCT

Even without segmentation, structures are visible

Different levels of enhancement

Surgical Training

Images by Nik Svakhine

5

Illustrative Rendering and Visualization on Mobile Devices

Challenges:

- *Limited battery, network, memory*
- *Limited screen resolution*
- *Limited graphics APIs*

Adaptation to Mobile Applications

Other constraints

- Often time-critical environments
- Simple interfaces and interaction key
- Intuitive design important
- Abstraction to key components for task critical

Example Applications

Maintenance and repair of aircraft

Emergency response

- Increased situational awareness
- In-field investigation

Illustrative Visualization on PDAs - Results
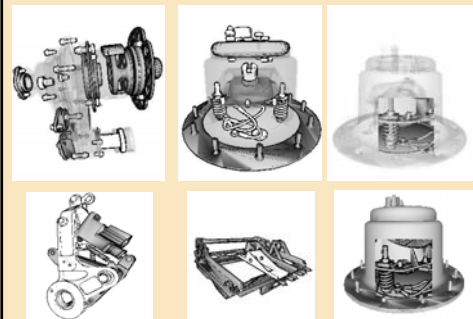
Images by
Jingshu Huang

More Illustrative PDA Visualization

Images by Jingshu Huang

Utility of Volume Illustration

Enhancing presentation

- Teaching
- Explaining
- Convincing

Reinforcing unreality

Emphasizing important features

Conclusions

Volume illustration is an effective, powerful tool !

- Effective enhancement / extraction of information
- Perception research
- Art / illustration techniques
- Interactive

Conclusions

Visualization is most powerful when combined with

- Effective enhancement / extraction of information
- Perception research
- Advanced illumination and shading
- Art / illustration techniques
- Improved interaction
- A larger solution

Acknowledgments

Collaborators:

- Aidong Lu, Nikolai Svakhine, Chuck Hansen, Chris Morris, Penny Rheingans, Elliot Fishman, Bill Oliver, Joe Taylor, Mark Hartner, Tim Thirion, Ross Maciejewski, Don Stredney, Mario Costa Sousa, Amy Gooch, Kelly Gaither, Yun Jang, Brian Bue, Jingshu Huang

Funding:

- National Science Foundation: NSF ACI-0081581, NSF ACI-0121288, NSFIIS-0098443, NSF ACI-9978032, NSF MRI-9977218, NSF ACR-9978099
- Air Force Research Lab
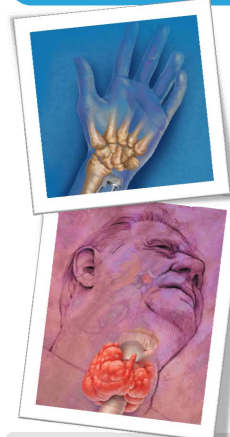- Adobe
- Nvidia
- Department of Homeland Security

Interactive Illustrative Rendering with Style

Stefan Bruckner

Institute of Computer Graphics and Algorithms

Vienna University of Technology

Illustrations

- An illustration is a picture with a communicative intent
- Conveys complex structures or procedures in an easily understandable way
- Illustrations use abstraction to prevent visual overload
- Abstraction allows the viewer to focus on essential aspects without losing context

---

Direct Volume Illustration

- Detailed volume data is readily available (medicine, biology, etc.)
- Illustrator's research process is significantly shortened
- Possibility to easily explore different stylistic choices
- Customized illustrations depicting particular pathologies
- Static illustrations, animations, interactive illustrations

---

Abstraction (1)

- Fundamental for creating an expressive illustration
- Introduces a distortion between visualization and underlying model
- Different degrees of abstraction based on the intent of the illustration
- Task of an illustrator: choose and apply abstraction techniques

---

Abstraction (2)

- High-level abstraction: deals with **what** should be visible and recognizable

- **Low-level abstraction: concerned with how different objects are presented**

Style Representations

- A good representation for visual styles has to fulfill certain requirements
  - **Flexibility** – ability to represent many different rendering styles
  - **Compactness** – simple and intuitive representation
  - **Transferability** – easy extraction from existing artwork
  - **Efficiency** – little overhead during rendering to allow interactivity

---

Conventional Approach

- Transfer function augmented by various additional parameters
  - Light directions and colors, shininess, tone shading parameters, silhouette color and thickness, …
- Complex and potentially costly at runtime, particularly if data-dependent
- Parameters are difficult to obtain, much fine-tuning required

---

Lighting Maps (1) [Bruckner and Gröller 2005]

- Simple solution for various styles with constant costs at runtime
- Two-dimensional lighting map which takes two dot products (N.L, N.H) as arguments
- Defines lighting contribution of a sample in …
  - Object color (i.e. "diffuse" for Phong shading)
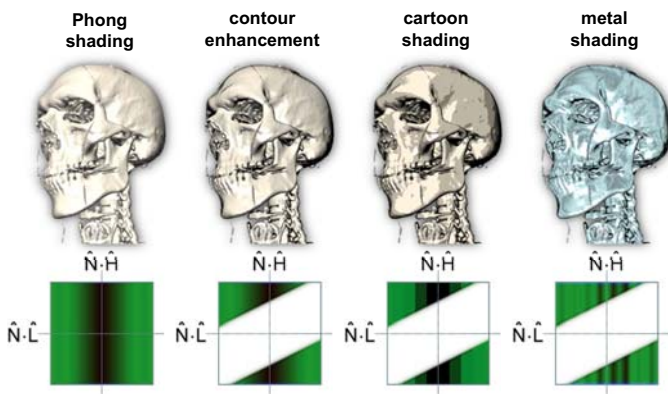 - Light color (i.e. "specular" for Phong shading)
  - Opacity

## Lighting Maps (2) [Bruckner and Gröller 2005]

- Lighting maps and their effects

Phong shading	contour enhancement	cartoon shading	metal shading

$\hat{N} \cdot \hat{H}$　$\hat{N} \cdot \hat{H}$　$\hat{N} \cdot \hat{H}$　$\hat{N} \cdot \hat{H}$

$\hat{N} \cdot \hat{L}$　$\hat{N} \cdot \hat{L}$　$\hat{N} \cdot \hat{L}$　$\hat{N} \cdot \hat{L}$

## Lighting Maps (3) [Bruckner and Gröller 2005]

- Simple representation which allows shading at constant costs
- Limited flexibility as color effects are not included in the map
- Still requires additional parameters (e.g., light position and color)
- Somewhat unintuitive for the user, not easily editable

Lit Sphere Maps (1) [Sloan et al. 2001]

- Use an image of a sphere under orthographic projection to shade another object
- Like environment mapping, but eye-space normal is used instead of reflection vector
- Light sources appear to be fixed to the camera
- Flexible image-based illumination, captures many different rendering styles

---

Lit Sphere Maps (2) [Sloan et al. 2001]

- Use a sphere map indexed by the eye-space normal to determine the color of a point


Lit Sphere Maps (3) [Sloan et al. 2001]
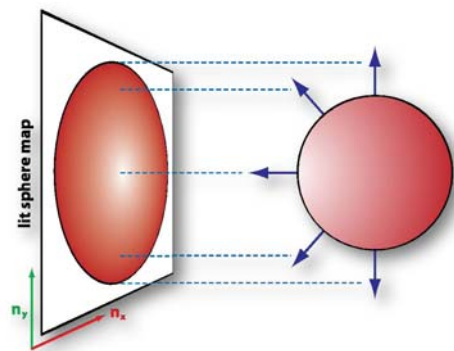
- Easy to obtain – lighting studies are frequently performed using spheres
- Sloan et al. describe simple extraction process from existing works of art
- Intuitive representation, can be directly displayed to the user as a preview

---

Lighting Transfer Functions [Lum and Ma 2004]
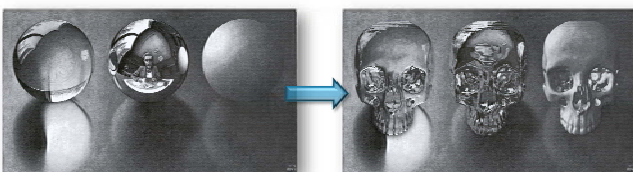
- Data-driven shading – illumination settings vary based on the data value
- Transfer function stores shading parameters in addition to colors and opacities

---

Style Transfer Functions (1) [Bruckner and Gröller 2007]
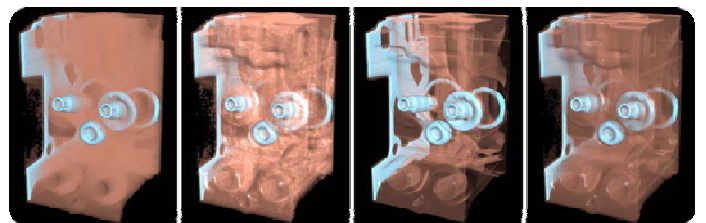
- A style representation allows us to shade one object in a given style
- For volume data, we rarely have discrete objects
- We need a continuous parameterization of style space
- A style transfer function maps volumetric attributes to visual styles

---

Style Transfer Functions (2) [Bruckner and Gröller 2007]

0 　　　　　　　　　　　　　　　　　　　1
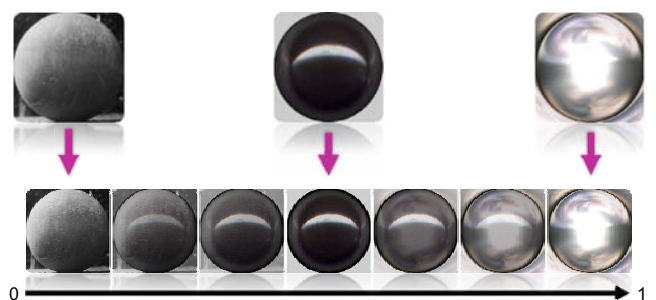
Style Transfer Functions (3) [Bruckner and Gröller 2007]

- Use lit sphere maps to enable data-dependent illustrative shading for volume rendering
- One lit sphere maps represents one specific rendering style
- Transfer function is defined over styles instead of colors
- Combines the power of data-dependent lighting with the flexibility of lit sphere maps

Stefan Bruckner   16

Style Transfer Functions (4) [Bruckner and Gröller 2007]
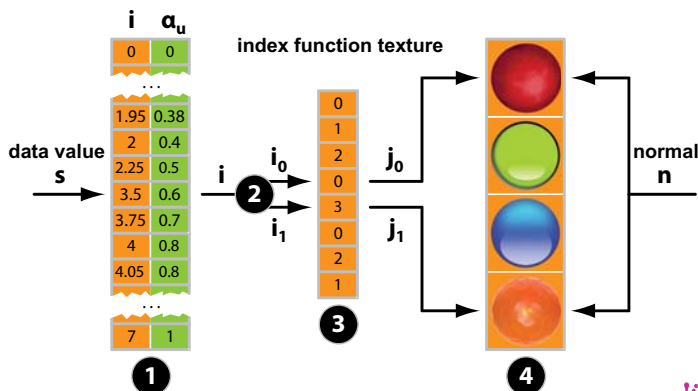
Regular Transfer Function   **Style Transfer Function**

Style Transfer Functions (5) [Bruckner and Gröller 2007]

transfer function texture   style function texture

index function texture

data value s → i → **2**

Stefan Bruckner   18

Style Transfer Functions (6) [Bruckner and Gröller 2007]

- Style transfer functions allow for a flexible combination of different visual styles


Stefan Bruckner

Style Contours (1)

- Contours are a frequent stylistic element in illustrations
- Contour appearance should be derived from lit sphere map
- Apparent contour thickness varies based on curvature
- Solution by [Kindlmann et al. 2003]: use normal curvature along the view direction to modulate contour threshold

Stefan Bruckner   20

Style Contours (2)

- Kindlmann's approach requires expensive reconstruction of 2nd order derivatives
- Simple approximation: angle between the gradient direction at two subsequent sample locations along a ray divided by step size
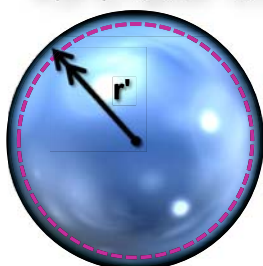

viewing ray

Stefan Bruckner   21
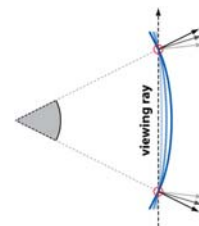
Style Contours (3)

- Instead of simple threshold, push lit sphere lookup coordinates outwards along the radius based on fuzzy "contourness" criterion

$$\text{if } |n \cdot v| \leq \sqrt{k_v(2 - k_v)}$$

$$\delta = 1 - \min(1, \frac{\sqrt{k_v(2 - k_v)} - |n \cdot v|}{\sqrt{k_v(2 - k_v)}})$$

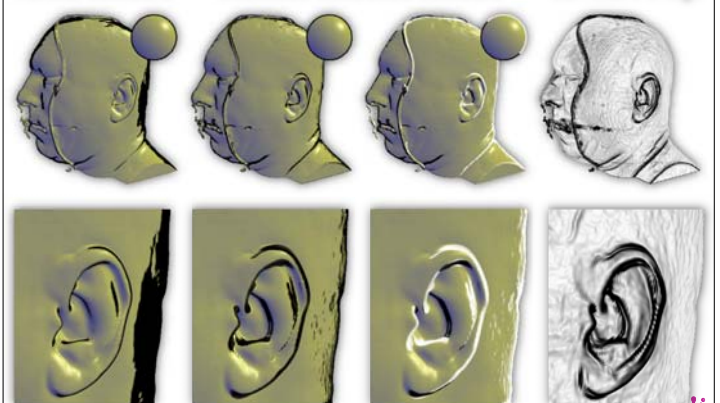$$r' = \min(1, \frac{r}{\delta})$$

Stefan Bruckner   22

Style Contours (4)

normal contours   **thickness-controlled contours**   **curvature image**

Stefan Bruckner   23

Style Contours (5)

Semantic Layers (1) [Rautek et al. 2007]
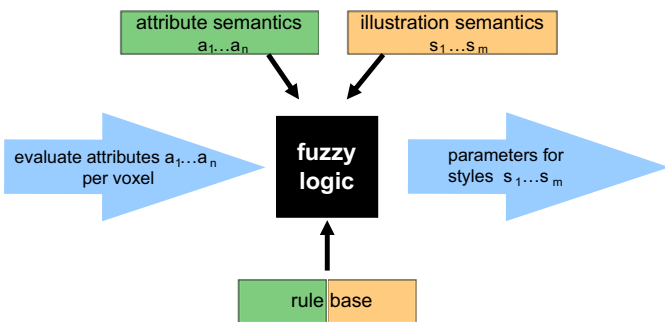
- Specifying a mapping for many different attributes is difficult
  - ◆ if principal curvature is negative
    then contour style is blue
  - ◆ if principal curvature is negative and density is high
    then contour style is blue
  - ◆ if principal curvature is negative and density is high and gradient magnitude is high
    then contour style is blue
  - ◆ if (principal curvature is negative and density is high and gradient magnitude is high) or …
    then contour style is blue
  - ◆ …

Semantic Layers (2) [Rautek et al. 2007]

- Use semantic rules to specify mapping from multiple volumetric attributes to multiple styles

volumetric attributes

density:
low – … – high

curvature:
negative – zero – positive

etc.

illustrative styles

contour style

transparent
subtle black
black
dark red
red
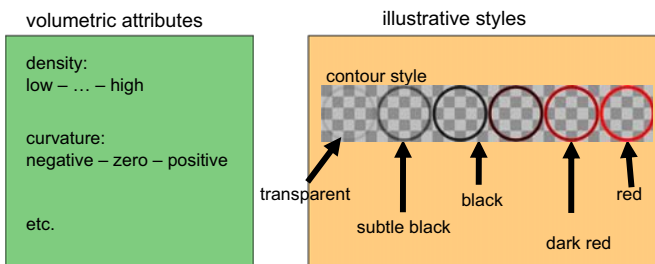
rules: if attribute a1 is v_{a1} … then style s1 is v_{s1}


Semantic Layers (3) [Rautek et al. 2007]

- Application semantics and semantics for visual styles are separated
- Membership functions are defined for volumetric attributes and visual styles
- Linguistic rules are used to specify the mapping from attributes to styles
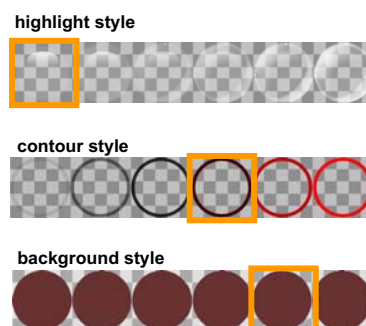- Fuzzy logic is employed for the evaluation of these rules

---

Semantic Layers (4) [Rautek et al. 2007]

attribute semantics $a_1 … a_n$

illustration semantics $s_1 … s_m$

evaluate attributes $a_1 … a_n$ per voxel

fuzzy logic

parameters for styles $s_1 … s_m$

rule base

Membership Functions

- if-part: semantics for volume attributes

membership

semantic value

negative
close to zero
positive

attribute   curvature

- then-part: semantics for visual appearance

membership

transparent   black   red

contour style

Semantic Rules

close to zero
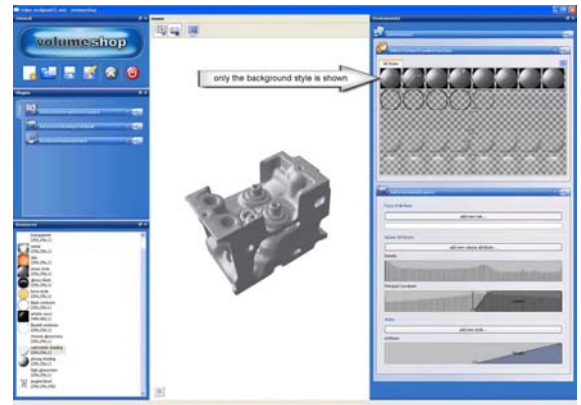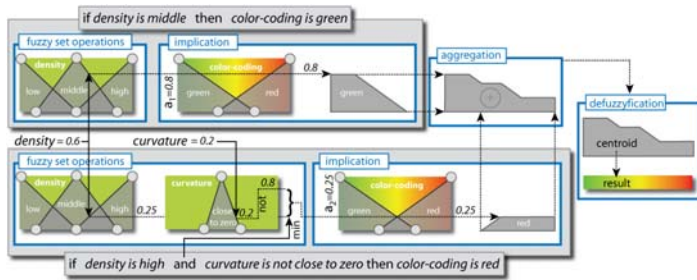
negative   positive

principal curvature
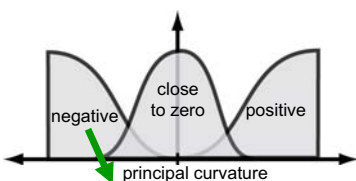
- if (principal curvature is negative and density is high and gradient magnitude is high) or distance to user focus is low
  then contour style is red

transparent   black   red

contour style
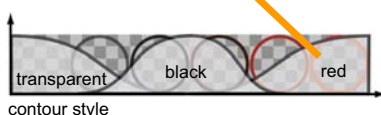
Layered Styles

- Apply styles in layers on top of each other, similar to the way illustrators work

highlight style

contour style

background style

Fuzzy Logic Evaluation

- For each style, all modifying rules are evaluated, aggregated, and defuzzified

if density is middle then color-coding is green

if density is high and curvature is not close to zero then color-coding is red
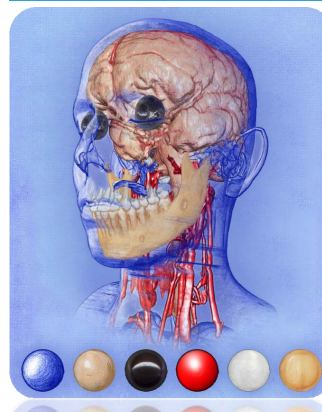
Demonstration

Conclusions

- A unified style representation extends the flexibility of illustrative shading
- Style parameterization allows mapping of volumetric attributes to visual styles
- Semantic layers provide an interface for specifying this mapping
- Layered styles enable the mapping of independent variables

http://www.volumeshop.org

Thank you for your attention! Questions?

References

- **[Bruckner and Gröller 2005]** S. Bruckner, M. E. Gröller. VolumeShop: An Interactive System for Direct Volume Illustration. *Proc. IEEE Visualization 2005*. pp. 671-678.
- **[Sloan et al. 2001]** P.-P. Sloan, W. Martin, A. Gooch, B. Gooch. The Lit Sphere: A Model for Capturing NPR Shading from Art. *Proc. Graphics Interface 2001*. pp. 143-150.
- **[Lum and Ma 2004]** E. S. Lum, K.-L. Ma. Lighting Transfer Functions Using Gradient Aligned Sampling. *Proc. IEEE Visualization 2004*. pp. 289-296.
- **[Bruckner and Gröller 2007]** S. Bruckner, M. E. Gröller. Style Transfer Functions for Illustrative Volume Rendering. *Proc. Eurographics 2007*. pp. 715-724.
- **[Rautek et al. 2007]** P. Rautek, S. Bruckner, M. E. Gröller. Semantic Layers for Illustrative Volume Rendering. *Proc. IEEE Visualization 2007*. pp. 1336-1343.
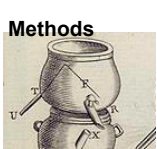
Intuitive and Ergonomic Interaction in Illustrative Visualization

Ivan Viola

UNIVERSITETET I BERGEN

Purpose of Illustration

Methods

Induction

Illustration for

Classification

Concepts

Observation

Ivan Viola — 1

Origins of Illustrative Visualization
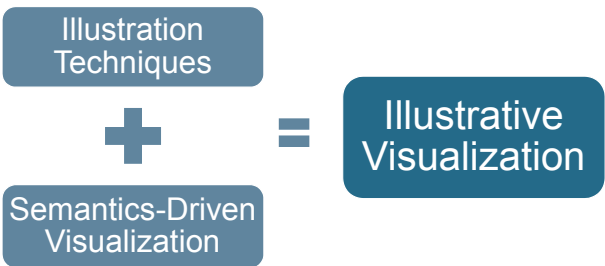
Illustration Techniques **+** Semantics-Driven Visualization **=** Illustrative Visualization

Ivan Viola — 2

Media and Styles in Illustration
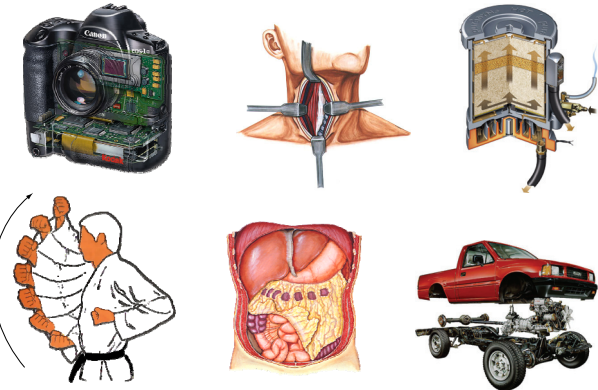
Ivan Viola — 3

Rendering styles in Visualization

Ivan Viola — 4

Expressive Techniques in Illustration

Ivan Viola — 5

Expressive Techniques in Visualization

Ivan Viola — 6

Semantics-Driven Visualization

Abstraction Levels

Information
Objects in Relation
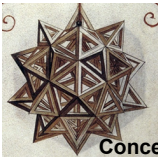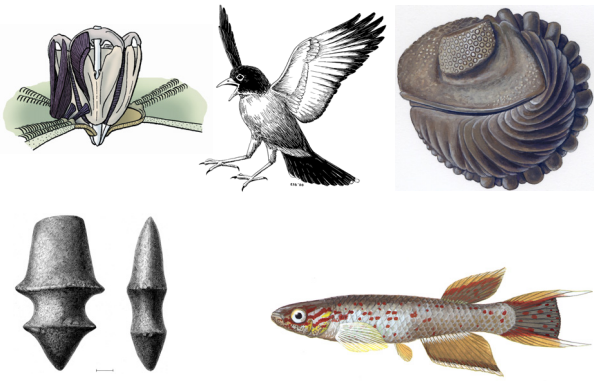Objects
Measurements

Visual Mapping
Relevance
Segmentation

Ivan Viola — 7

Semantic Layers in Visualization

[Rautek et al. 2007]

value for style
low — high

Rule 1
if *density is high* then *style S₀ is high*

Rule 2
if *principal curvature is not close to zero* then *style S₁ is high*

if *density is high* then contours are thick and tissue style is bone
if *distance to vessels is low* then color style is yellow
if *distance to vessels is very low* then color style is red

if *principal curvature is not positive* then contours are blueish

Ivan Viola and Peter Rautek — 8

Focus+Context Visualization

- Goal: visual focus–context discrimination
- Degree of interest (DOI) function
- Uneven use of graphics resources

Opacity   Style   Color   Frequency   Space

lists   graphs   visualization   **F+C visualization**

0 1 2 3 4 5 6 7 8 9 $\log_{10}$(#items)

Ivan Viola and Helwig Hauser — 9

Importance-Driven Visualization Model

importance specification

levels of sparseness

dense
representation
0   importance   max

Ivan Viola — 10

Importance-Driven Visualization Model

Abstraction Levels

Information
Objects in Relation
Objects
Measurements

Visual Mapping
Relevance
Segmentation

Ivan Viola — 11

Ghosting Cutaways

Object Categories

Importance

Focus

Close Relation to Focus

Context

Example – Abdominal Structures

Lung Nodules Visualization

Visualization of MR Mammograms

[Coto et al. '05]
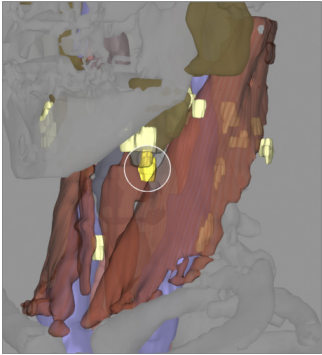
VesselGlyph – Angiography Visualization

[Straka et al. '04]
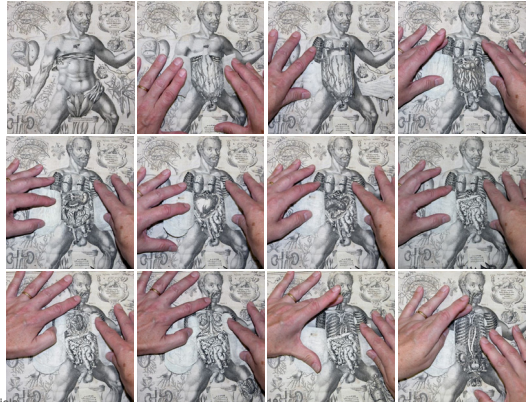
Neck Dissection Planning

[Krüger et al. '05]
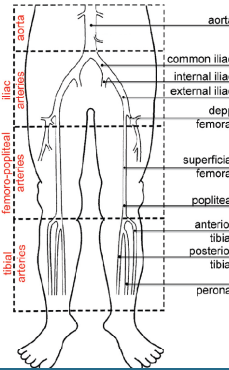
Coupling Illustration and Interaction

Visualization Yes ! – Interaction No ?

- Problems
  - Interaction is very time-consuming
  - Interaction prevents comparisons
  - Interaction hampers reporting
- Challenges
  - Provide standardized views
  - Algorithms highly parameterized – provide sensible default settings
  - Support automatic parameter tuning
  - Provide navigational aids
- Examples
  - Automatic view point selection
  - Focus of attention
  - Automatic light placement (inconsistent lighting)
  - Automatic reporting
  - Dynamice poster - automatic storytelling
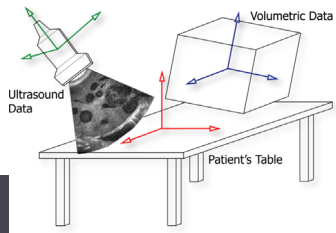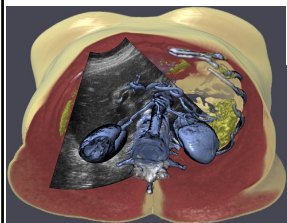
Quoting Master from NorVis07
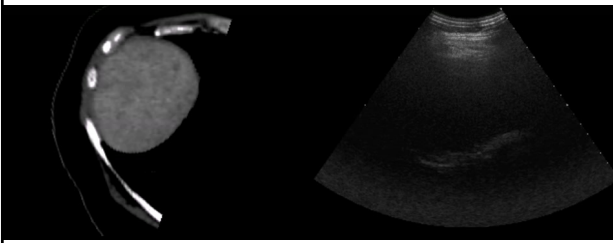
Contextual Cutaways for MultiModal Vis

[Burns et al. '07]
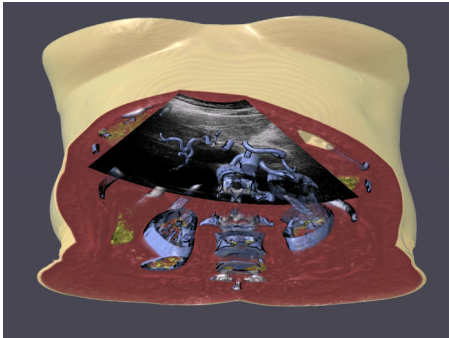
Sync Views of Co-Registered CT+US

CT Scan Data   Ultrasound Data

Fused Modalities Using Cut-Aways

Illustrated Ultrasound

LiveSync of 2D-3D Views for Exploration

[Kohlmann et al. 2007]

Guided Navigation

- Input: known and classified volumetric data
- High level request: show me object X
- Output: guided navigation to object X

Interactive Focus of Attention

Example - Stagbeetle

Focus view 1

Overview   Focus view 2

Example - Human Hand
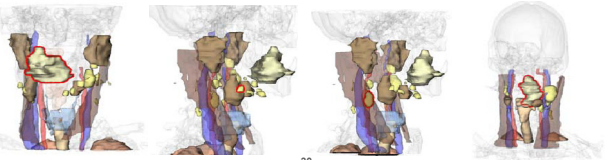
Application-Driven View Selection

[Mühler et al. 2007]

- Emphasis on the domain knowledge
- Distance to important feature defines importance of other features
- Shortest path on the bounding spheres
- Zooming to focus
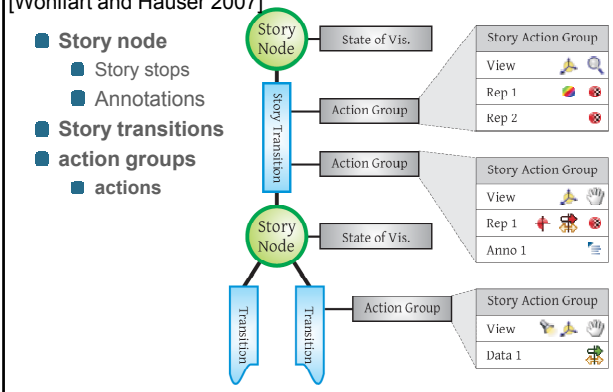
Storytelling for Presentation
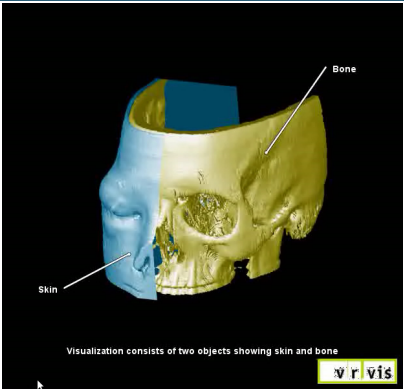
[Wohlfart and Hauser 2007]

- **Story node**
  - Story stops
  - Annotations
- **Story transitions**
- **action groups**
  - **actions**
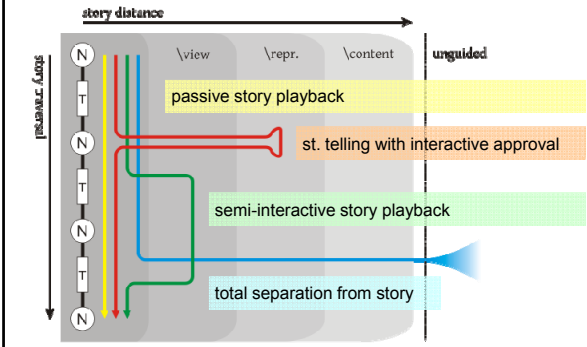
Sample Story

Interaction Patterns

Purpose of Illustrative Visualization

Selective Visualization ← Illustrative Visualization → Interactive Exploration → Interactive Presentation

---
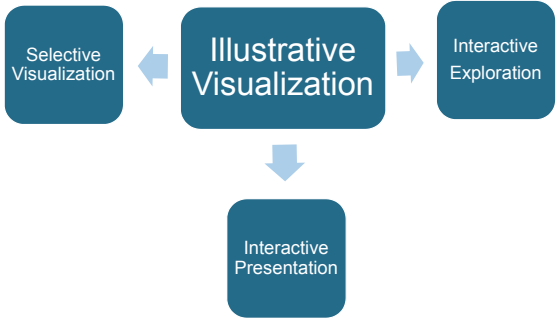
Also featuring

This presentation includes slides, videos and images of:

- Meister Edi Gröller
- Helwig Hauser
- Stefan Bruckner
- Peter Rautek
- Konrad Mühler
- Katja Bühler
- Mario Costa Sousa
- Ernesto Coto
- Arno Krüger
- Matúš Straka
- Daniel Patel

- Timo Ropinski
- Michael Wohlfart
- Carlos Correa
- Kevin Hulsey
- MediGraphics

Thank you!

Sketch-based Volumetric Seeded Region Growing

H. L. J. Chen[1]   F. F. Samavati[1]   M. C. Sousa[1]   J. R. Mitchell[1,2] [†]

[1]Department of Computer Science, University of Calgary, Canada
[2]Seaman Family MR Research Centre, Foothills Medical Centre, Calgary, Canada [‡]

Abstract

Interactive volume segmentation is an essential and important step in medical image processing. Conventional interactive methods typically demand significant amounts of time and do not lend to a natural interaction scheme with the 3D volume. In this paper we present a sketch-based interface for seeded region growing volume segmentation. In our approach, the user freely sketches regions of interest (ROI) directly over the 3D volume. Parts of the volume outside the ROIs are then automatically cut out in real-time. The user repeats this process as many times as necessary until he/she decides to specify the seed point 3D location directly at the ROI. To prevent unexpected segmentations, the region growing is restricted to the specified ROI. Our sketch-based system utilizes GPU programming to achieve real-time processing for both rendering and volumetric cutting independent from the size and shape of the sketched strokes.

Categories and Subject Descriptors (according to ACM CCS): I.4.6 [Image Processing and Computer Vision]: Segmentation, partitioning

1. Introduction

Medical imaging systems, such as computerized tomography (CT), magnetic resonance imaging (MRI) and ultrasound, are becoming increasingly ubiquitous. Clinicians and surgeons often use computer-based segmentation to identify and analyze anatomical structures of interest in medical image datasets. For example, neuroradiologists often segment and examine the internal carotid artery to determine its degree of stenosis in patients suffering from transient ischemic attacks (TIAs - "mini" strokes). The degree of carotid stenosis is a critical factor to determine if TIA patients should have surgery to open up this vital vessel. Other measurements (such as the shape, topology, and cubic volume) could also be obtained during the segmentation process [ONI05]. Therefore, volume segmentation is an essential and important step in medical image processing.

Segmentation is often broken down into "edge based" or "region based" methods. Each of these in turn may be "manual" or "computer assisted" (including completely automatic). Along the edge-based category, a typical manual segmentation process requires a trained specialist to draw contours around the region of interest (ROI) on cross-sectional images. These contour lines are then linked and reconstructed into a 3D representation for further analysis (Figure 1, top). This procedure can become a challenging task if the target is, for example, blood vessels in the brain, which by nature involves complex shape and unpredicted turning directions. Automatic methods currently focus on low-level features such as edge detection and texture analysis. An example of an edge detection algorithm exists in the use of histograms by considering the relationship between three quantities: the data value and its first and second directional derivatives along the gradient direction [KD98]. A number of contributions and efforts were made in the research direction for obtaining automatic segmentation results. However, the difficulty for a complete automatic approach is limited in one sense or another. Kirbas and Quek [KQ03] pointed out that all such attempts for developing automatic segmentation algorithms are limited to some global parameters or can fail with certain data.

The region growing [RK82] algorithm is one of the well-

[†] http://www.ImagingInformatics.ca
[‡] http://www.mrcentre.ca
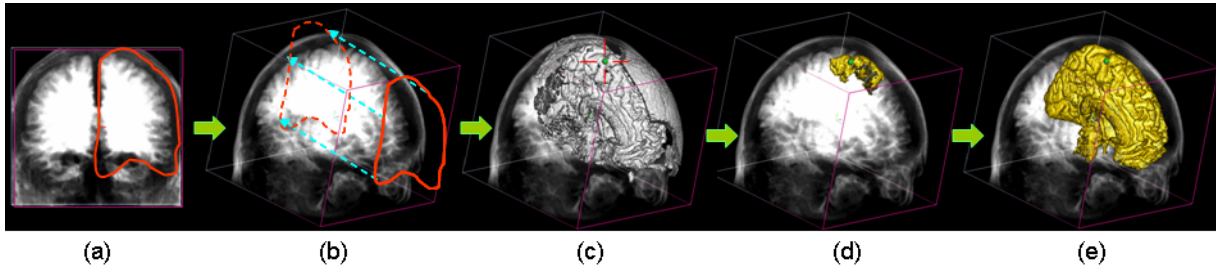
Figure 2: *Our sketch-based volume segmentation method: user sketches a ROI directly over the data (a), the ROI is extruded (b), volume outside is cut out and user plants the seed point (c), region grows (d) and segments volume portions within the extruded ROI (e).*
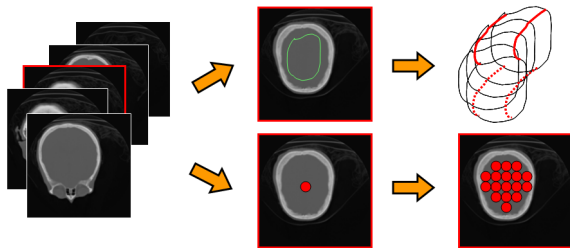
Figure 1: *Conventional segmentation methods. Top row: edge-based method. Bottom row: region-based method.*

known region-based segmentation methods that is simple to compute and applicable to a wide range of data types. Seeded region growing was first introduced by Rolf Adams and Leanne Bischof [AB94]. Their algorithm requires the planting of an initial seed point in the 3D volume dataset. However, the challenge of specifying a 3D coordinate from a 2D device, such as the mouse, is associated with providing an intuitive interface in assisting with the mapping process. Existing methods for specifying the seed point [SHN03] can be outlined as follows: the user navigates from a stack of 2D image slices; a desired slice is selected (i.e. equivalent to selecting one of the axis as a first step); and then the user places the seed point from the cross-sectional view of the data (Figure 1, bottom). As a result the seed point is propagated to the entire volume based on certain criteria the user defines.

The key limitations with the conventional seeded growing region process are the large amount of cross-sectional images a user has to go through. The user is also required to have a priori knowledge of the data in order to quickly identity the correct slice number and the appropriate seed location on the 2D grey-scaled image. This procedure demands a significant amount of time and does not lend to a natural interaction scheme with the 3D volume (i.e. direct manipulation of the 3D data).

In this paper, we propose a sketch-based interface for volumetric seeded region segmentation. Figure 2 illustrates the

key stages of our method applied over a raw MRI super-brain dataset (152x154x181). At first, the user loads the volumetric data and defines an intensity range from the histogram. And then the user directly sketches a ROI over the displayed volume (Fig. 2, a). The system extrudes the ROI along the viewing direction within the entire volume (Fig. 2, b - dotted lines). The volume outside the extruded ROI is cut out and the user places the seed at the red cross (Fig. 2, c). The region starts to grow (Fig. 2, d) and finally the complete segmentation inside the extruded ROI is obtained (Fig. 2, e). In addition, the user could place multiple sketches from different views to form arbitrary-shaped ROI. Our system uses GPU programming for real-time rendering and interactive sketching. Furthermore, we utilize the stencil buffer to achieve a processing rate that is independent of the sketch complexity.

The rest of the paper is organized as follows. In Section 2, we review related work and current sketch-based interfaces for volume segmentation. In Section 3, we outline our system framework. In Sections 4, 5, and 6, we provide details of our sketch-based system for volume segmentation. Results are discussed in Section 7, and conclusions are presented in Section 8.

2. Related Work

Interactive seeded region growing. Many segmentation approaches have been proposed for the 2D image segmentation task. The set of well-known techniques include thresholding, k-means clustering, watershed segmentation, and level-set methods (see the survey conducted by Pham et. al. [PXP99]). For segmenting 3D medical datasets, these techniques could also be applied and adapted easily by re-using the 2D image techniques. Sherbondy et. al. [SHN03] developed a fast volume segmentation system using GPU. Their work was based on seeded region growing. The seed selection step allows the user to paint seeds by drawing on the sectional views of the volume. Their segmentation merging criteria are based on non-linear diffusion metric. They also incorporated image smoothing algorithms for noise conditions. More recently, Schenke et. al. [SWD05] analyzed the GPGPU paradigm

and implemented the seeded region growing method with fragment shaders and VTK. In order to fully take advantage of the GPU parallelism, the user was encouraged to specify as many seed points as possible.

Sketch-based interfaces for volume segmentation. For general sketch-based modeling of volumetric data, Owada et. al. [ONNI03] presented a system that captures hand-drawn sketches and creates volumetric objects with internal structures. Owada et. al. [ONOI04] further extended the interface for users to define internal volumetric textures of a model. The system allowed interactive design and browsing for volumetric illustrations. Recent work for segmenting volumetric data have also focused on incorporating user intervention and developing interactive segmentation systems. Tzeng et. al. [TLM03] developed a novel interface for volume data classification. They allowed the user to draw strokes on the cross-section of volume data that roughly indicate foreground and background regions. The stroke information was used to train a classifier that is designed for segmenting voxels. Yuan et. al. [YZNC05] presented a novel method to cut out volumetric structures by drawing simple strokes directly on volume rendered images. Owada et. al. [ONI05] proposed an intuitive user interface for volume segmentation. The user traces the contour of the target region using a 2D free-form stroke on the screen. The volume catcher system then returns a plausible 3D region inside the stroke.

Similar to Owada's approach [ONI05], the concept of our system extends the stroke and sweeps through the volume. We use histograms as a first classification step whereas they applied opacity transfer functions. In contrast, we adapt closed strokes that include free-form and other variations. Most importantly, our approach allows the user to interact with a simple sketch-based interface for navigating to the ROI instead of browsing through hundreds of cross-sectional slices ([SHN03]; [SWD05]). For seed planting, our technique is fundamentally 3D and the user no longer needs to look at texture-mapped 2D planes. In addition, we enable the user to define a sub-volume of arbitrary shape with few sketches to constrain the region grow and provide rapid segmentation feedback.

3. Particle System Framework

In our sketch-based system, we utilize a particle system framework. Because of the generality and the fundamental design of the framework, the system can be easily extended to work with irregular datasets. Other potential applications include general point-based systems and polygonal meshes (which were converted to a point-cloud).

At the first stage of our system, a desirable range of intensities is selected by using the intensity histogram to define target voxels from the volumetric dataset. Since only a subset of the entire volume is rendered to the scene, we represent the target voxels by a particle system. We use lists of particles for rendering and processing. This avoids the need to traverse the 3D array containing the original dataset every time we access these target voxels.

In order to maintain the lists of particles, we organize them with a central particle system scheme. The particle system contains a list of particle objects. Each particle object can be organized and displayed by using the display list or vertex buffer objects (VBOs). When the display list option is used, each particle object contains an object color if particles do not possess color information. The particle object also maintains a list of particles and each particle contains information such as: position (x, y, z), color (r, g, b, a), and reference to voxel (which contains intensity and gradient). Position is used during the sketch-based volume cutting (Section 4). Reference to voxel is required to locate neighboring voxels during segmentation (Section 5). For rendering (Section 6), position, color, and voxel gradient (normal) are needed.

Alternatively, particle objects can utilize the various VBOs stored in a collection of *particle buffersets*. Each *particle bufferset* contains a vertex buffer (i.e. voxel position), normal buffer (i.e. voxel gradient), and color buffer (i.e. voxel intensity). Each of these buffers is stored on the GPU texture memory using VBO. The required voxels only travel across the system bus once whenever the histogram is defined. Each particle object then maintains only index information into the corresponding *particle bufferset*. Particles are rendered in either X-ray mode or surface mode (Section 6). Each particle object contains an attribute for its assigned rendering mode.

4. Sketch-based Volume Cutting

To place the seed for the region growing, we use a novel sketch-based interface. In the first stage, the user specifies a ROI by a closed free-form sketch on the screen. The extrusion of this sketch forms the ROI and likely contains the target area (organ). This approach has several benefits: it increases the performance of seed-growing after extrusion and cutting, the user is able to navigate and place the seed more easily, and finally it is very intuitive.

The main challenge here is to cut the extrusion from the volume at an interactive rate. With the defined histogram intensity range, a collection of particles is composed from the 3D volume array. The set of particle attributes is packaged into vertex buffer, normal buffer, and color buffer using VBO. These buffers are sent only once and stored on the GPU texture memory for successive rendering and processing. The sketched area is extruded along the view direction and pierces into the entire volume (Figure 3). The computed sub-volume is rendered in the surface mode and the background volume is rendered in the X-ray mode (Figure 9, right). Subsequent sketches affect only the 'visible' sub-volume currently rendered in the surface mode. Then the remaining task is to distinguish the particles that fall 'inside' the extrusion from the ones that are 'outside'.
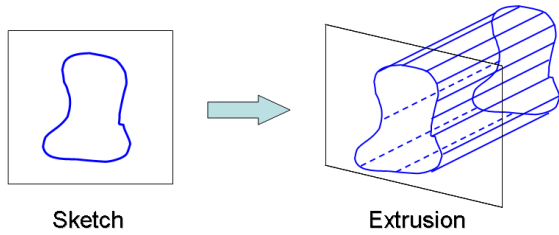
Figure 3: *Sketch extrusion.*

color [WNDS99]. Then we save the content of the stencil buffer as a texture as demonstrated in Figure 5.

Figure 5: *Generating the computational mask using the stencil buffer.*

In order to find the list of selected particles that fall inside the sketched extrusion, one possible strategy is to use the standard polygon fill or crossing test algorithms [Hai94] [Fra]. For this, we can project the particle to the screen and check whether it is inside of the sketched stroke. The speed of this method depends on the number of points on the stroke (as a polygon). Unfortunately, this method suffers from a slow speed when the stroke (polygon) has a good quality. Although we could implement the crossing test in GPU, the speed is still dependent on the complexity of strokes and the level of interactivity can vary depending on the user input. Instead, we adapt a novel GPU-based technique that is independent of the sketch complexity.

4.1. Computational Mask

The fundamental concept of our sketch-based system is a real-time filtering process employing a computational mask (Figure 4). We move the mask to traverse the entire volume in a front-to-back order and pick up the particles (or voxels) that are inside the sketched area. The particles which are not visited by this process are labeled as being 'outside'. As depicted in figure 4, the volumetric dataset can be in any orientation with respect to the computational mask.
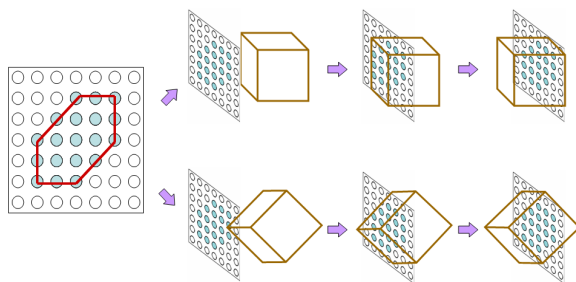

Figure 4: *Computational mask.*

The computational mask is composed of 1s and 0s, where 1 indicates that the pixel is covered by the sketched area and 0 means that the pixel is outside the area. Figure 5 illustrates the process for generating the mask. At first, the user places strokes on the screen and a closed curve is obtained. Next, we fill the enclosing area using the stencil buffer with a 1-bit

4.2. GPU-based Sketch System

In order to quickly filter the entire volume with the generated computational mask, we perform all of our computation in GPU and send back the result as a single texture to the CPU. We leverage the workload to both vertex and fragment shaders and optimize the speed by minimizing the program complexity. Notice that we use the fragments (pixels) and the framebuffer somehow different from their regular functions. Instead of sequential processing of the particles, we map many particles to the fragments at a time. This helps us to use parallel architecture of GPU for processing of the particles. Therefore, the fragments' "position" in our method is an index to the particles instead of being a position of visible pixels. We also use a binary "value" for the fragments to show whether the particle is inside of the extrusion. To map the index of particles, which has a linear order, to the position of fragments, that has two components, we use a 2D texture coordinate buffer. In addition, not all particles can be uniquely mapped to the screen. Consequently, to process all of the particles, we need to do the process in several passes of saving the current screen and mapping a new set of particles. For saving the current screen, which contains binary values, we use a one-bit plane of the framebuffer (off-screen). For example, with a given 100x100 sized screen, we are able to process 10,000 particles for each pass through the graphics pipeline. Figure 6 gives an overview of the processing pipeline.

4.2.1. Preparing Data Buffers for Pipeline Processing

The vertex buffer (1) contains all particles collected from the histogram pre-classification phase. It is not deleted unless the intensity range has been redefined. This enables the system to quickly fetch the target particles whenever a sketched region shall be resolved. This mechanism prevents unnecessary traffic of particles traveling across the system bus for every processing cycle. The texture coordinate buffer (2) stores a 2D array of screen coordinates (0, 0), (0, 1), ..., (s, t), ..., (height - 1, width - 1). Each particle is mapped to a screen coordinate using the associated texture coordinate. During the execution of the processing pipeline, we redirect every particle to its designated screen location.
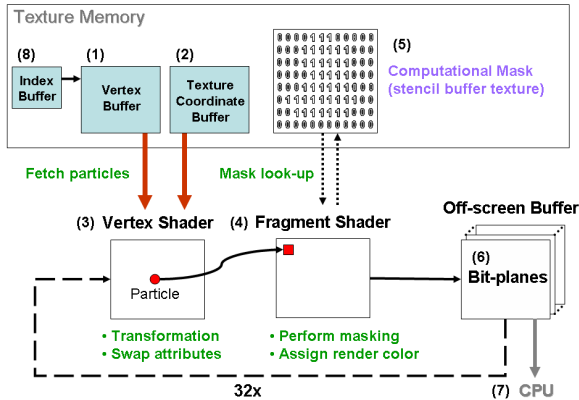
Figure 6: *Sketch system implemented in GPU.*

4.2.2. The Vertex and Fragment Programs

The vertex shader (3) is used to perform the particle coordinate transformation. In order to rasterize the current vertex (e.g. rendered as a 3D point) to the designated fragment location, we swap the incoming attributes as follows. The vertex (input) is multiplied by the model-view matrix, and the result is assigned to the texture coordinate (output). The accompanied texture coordinate (input) is multiplied by the projection matrix, and the result is assigned to the position (output). After the vertex shader has finished processing, both the resulting texture coordinate and position are rasterized and passed onto the fragment shader.

The fragment shader (4) performs the masking operation and assigns a pre-defined render color if the mask value is valid. The input texture coordinate (i.e. the particle's position assigned by the vertex shader) is adjusted with respect to the projection parameters and the value is looked up from the computational mask stored as a stencil buffer texture (5). If the texture look-up results a value of 1, then the particle processed by the current fragment program is inside the sketched region; otherwise, it is outside.

4.2.3. Parameter Calculations

Note that we only need one bit in the off-screen buffer (6) to store the selection information (i.e. one being selected, and zero being not selected). For a typical off-screen color buffer with RGBA components, and each component having 8-bit resolution, it is possible to encode 320,000 particle selections information by adding all render colors. Thus, the required off-screen buffer dimension is $\lceil \sqrt{N/32} \rceil$, where N is the total number of particles to be processed from the vertex buffer. The calculated buffer dimension becomes the width and height of the off-screen buffer.

4.2.4. Composing the Result

Finally, the CPU (7) receives the texture and decodes the selected particles to construct two index buffers, one con-

taining indices of the selected particles and the other one for the non-selected particles. These index buffers are then sent and stored in the GPU texture memory for the next processing cycle as well as for rendering purposes. In subsequent sketch operations, the index buffer (8) (storing the indices of the previously selected particles) is used to index into the vertex buffer when the 'fetch particles' command has been issued.

5. Seeded Region Growing

After describing a rough estimate of the target area using the sketch-based volume cutting, the user can navigate the volume and place a seed point directly on the visible surface to obtain an accurate segment. To find out the seed location in the 3D object-space from a 2D input device (e.g. the mouse), we use an intuitive interface that is consistent with our sketch-based system. In this interface, the user inputs a visible voxel (particle) by clicking the mouse on the screen. The entered pixel can be associated with several particles in various depths and we need to find the best candidate. To do this, we extend the pixel area to a larger rectangle whose extrusion in the volume contains all the involved particles (see Figure 7). To extrude the rectangle in the volume, we use the same technique as described in section 4. After determining all the involved particles, we select the one that has the shortest distance to the entered seed point (Figure 7). The selected particle is then used as the actual 3D seed point. For the region growing algorithm, we start from the seed point as the current voxel and move to adjacent voxels with intensity values close to the current intensity. We use the breath-first search algorithm as appears in the context of graph traversing techniques [CLRS01]. This approach helps to maintain a balanced and coherent growth. We use a threshold for the closeness of the intensities. It is obvious that the growing process can be sensitive to thresholds and the resulting region can be dramatically enlarged when the threshold is increased by one or two scales. However, as a benefit of our volume cutting tool, we can constrain the growing region to be inside of the cut sub-volume as a rough estimate of the desired region.
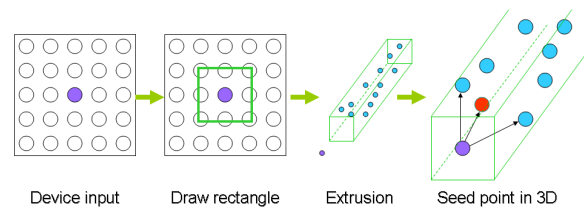

Figure 7: *Searching the seed point.*

6. Rendering

In our approach, we adapt the splatting technique [Wes91] for direct volume rendering using GPU programming. For

rendering the volumetric data, each particle associated with a voxel is rendered as a square texture using the OpenGL hardware accelerated point sprite. Point sprite enables us to send only a single vertex information for each particle (voxel) through the rendering pipeline. We adapt the Gaussian kernel as our texture generation function (Figure 8, left).

In the fragment shader, we simply check the incoming opacity value and discard the current fragment if alpha is less than 0.2. We adapt two rendering modes for point-based splatting: X-ray and surface modes (Figure 8, middle and right, respectively).
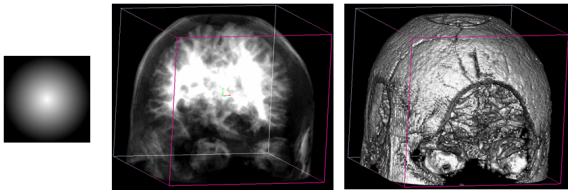
Figure 8: *(left to right) Disk texture with Gaussian distributed transparency values. Different rendering of the brain: X-ray and surface modes.*

The X-ray mode accumulates all fragments to compute the final pixel value with the following OpenGL formulation: $I_f(x) = \alpha_{new}(x)I_{new}(x) + I_f(x)$ [XC04]; where $I_f(x)$ is the frame-buffer intensity value at pixel location x, $I_{new}(x)$ is the incoming fragment value, and $\alpha_{new}(x)$ is the opacity of the new fragment. We use glBlendFunc(GL_SRC_ALPHA, GL_ONE) to perform the accumulation [XC04].

In order to render particles and obtain a surface representation, we apply a two-pass rendering technique that consists of the visibility pass followed by the shading pass [BHZK05]. During the visibility pass, we perform the so-called ε-test operation. For implementing the ε-test, we perform the following steps. First, we scale all the particles with the value of ε in the negative z-direction. Then we render to the depth buffer and turn off the color buffer. During the shading pass, we perform lighting computation for each particle processed by the vertex shader. Note that in both passes, we discard fragments whose opacities are less than 0.2. We also combine the X-ray mode and the surface mode to form the hybrid mode as follows: (1) render the particles (X-ray mode) to the frame buffer using alpha-blending with no lighting and (2) render the particles (surface mode) and perform the visibility pass and the shading pass respectively. However, during the visibility pass while rendering the surface mode particles, we enable writing to both the depth buffer and the color buffer. In the fragment shader, we output black pixels for all fragments processed (i.e. to overwrite the X-ray mode particles).
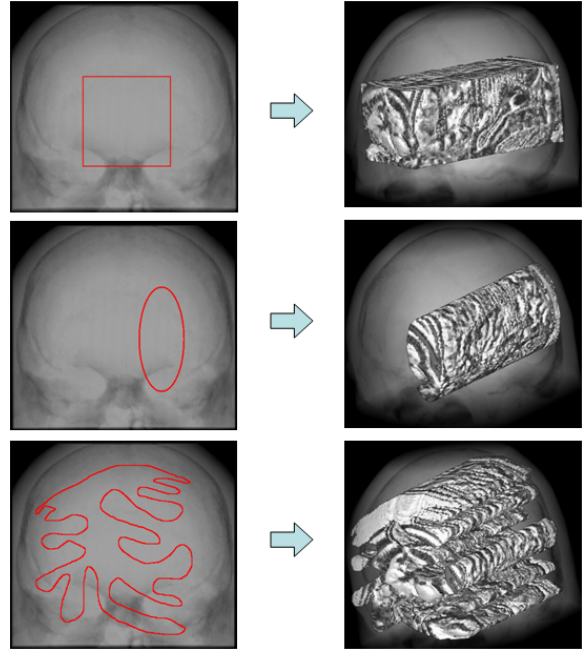
Figure 9: *Sketch types: (a) rectangular strokes, (b) elliptical strokes, and (c) free-form strokes.*

7. Results and Discussions

All the results were generated on an AMD Anthlon 64 X2 3800 with a GeForce 7800 GT, 256 MB card. We selected raw volumetric datasets of the brain (MRI, 152x154x181), skull (MRI, 256^3), and angiography (3T MRT, 256x320x128).

For all datasets, the sketch response time (SRT) was below 1 second. From loading a full-range histogram, Figure 2 shows the segmentation of grey and white matter of the left hemisphere of the brain (SRT = 0.384 sec). Figure 9 illustrates the before/after effects on the brain dataset after sketching rectangular, elliptical and free-form ROIs (SRT = 0.515, 0.392 and 0.384 sec, respectively). Figure 10 (top row) shows a successful segmentation of the right ventricle with SRT = 0.267 sec. Figure 10 (bottom row) illustrates a series of volume cutting after free-form sketched ROIs (SRT = 0.261 sec) and the resulting segmented portions of the teeth. With the 3T MRT time-of-flight angiography dataset of a human head (Figure 11), we were able to quickly segment the carotid and cerebral arteries with SRT = 0.224 sec.

Our system also allowed a real-time preview of seed locations as the user moves the mouse. The interactive rate of seed searching was achieved by utilizing the core system implementation and from the aid of GPU. Note that in order to obtain smooth sketching lines, we froze the background rendering (i.e. the volume splatting) by saving the entire scene to a texture. Thus when the user placed strokes on the screen,

we rendered the screen-sized texture first followed by the ROI strokes.

8. Conclusion and Future Work

We presented a novel interface for volume segmentation based on seeded region growing. Instead of the traditional way of browsing from hundreds of cross-sectional slices, we proposed a sketch-based interface for interactive volume exploration and navigation for the ROI. We provided real-time rendering when the user interacts and places the seed point from a truly 3D environment. More importantly, our sketch-based system constrained the region grow from the cut subvolume to enforce focus-of-attention. In designing from a particle system perspective, our approach can be easily extended to a number of applications including other point-based systems, polygonal meshes, and irregular volume with changing topology.

Future improvements include extending our system with other algorithms for sketch-based volume manipulation. It would also be useful to have the capability of multiple sketched ROIs assigned in different regions of the volume to allow, for instance, better control of the level of detail in selected regions of the dataset. The criteria that we used to judge the quality of the results were solely based on our observations on the speed and flexibility of volume data cutting, exploration, and seed planting/growing control. It is important to conduct more formal evaluations and user/clinical studies to provide quality sketch-based volume segmentation tools for professionals in medical science.

References

[AB94] ADAMS R., BISCHOF L.: Seeded region growing. *IEEE Trans. on PAMI 16*, 6 (June 1994), 641 – 647.

[BHZK05] BOTSCH M., HORNUNG A., ZWICKER M., KOBBELT L.: High-quality surface splatting on today's gpus. In *Proc. of the Eurographics Symposium on Point-Based Graphics '05* (2005).

[CLRS01] CORMEN T. H., LEISERSON C. E., RIVEST R. L., STEIN C.: *Introduction to Algorithms.* MIT Press and McGraw-Hill, 2001.

[Fra] FRANKLIN W. R.: Pnpoly - point inclusion in polygon test. http://http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html.

[Hai94] HAINES E.: Point in polygon strategies. *Graphics Gems IV* (1994), 24–46.

[KD98] KINDLMANN G., DURKIN J.: Semi-automatic generation of transfer functions for direct volume rendering: Methods and applications. In *Proc. of Visualization '98* (1998), pp. 79 – 86.

[KQ03] KIRBAS C., QUEK F.: Vessel extraction techniques and algorithms: a survey. In *Proc. of Bioinformatics and Bioengineering '03* (2003), pp. 238 – 245.

[ONI05] OWADA S., NIELSEN F., IGARASHI T.: Volume catcher. In *Proc. of the Symposium on Interactive 3D graphics and games '05* (2005), pp. 111 – 116.

[ONNI03] OWADA S., NIELSEN F., NAKAZAWA K., IGARASHI T.: A sketching interface for modeling the internal structures of 3d shapes. In *Proc. of 3rd International Symposium on Smart Graphics* (2003), pp. 49 – 57.

[ONOI04] OWADA S., NIELSEN F., OKABE M., IGARASHI T.: Volumetric illustration: Designing 3d models with internal textures. *Proceedings of ACM SIGGRAPH(SIGGRAPH2004)* (2004), 322–328.

[PXP99] PHAM D. L., XU C., PRINCE J. L.: A survey of current methods in medical image segmentation. *In Technical Report JHU/ECE 99-01, The Johns Hopkins University* (1999).

[RK82] ROSENFELD A., KAK A.: Digital picture processing. *New York Academic Press 2* (1982), 138 – 145.

[SHN03] SHERBONDY A., HOUSTON M., NAPEL S.: Fast volume segmentation with simultaneous visualization using programmable graphics hardware. In *Proc. of IEEE Visualization '03* (2003), pp. 171 – 176.

[SWD05] SCHENKE S., WUENSCHE B., DENZLER J.: Gpu-based volume segmentation. In *Proc. of IVCNZ '05* (2005), pp. 171 – 176.

[TLM03] TZENG F.-Y., LUM E. B., MA K.-L.: A novel interface for higher-dimensional classification of volume data. In *Proc. of IEEE Visualization '03* (2003), pp. 505 – 512.

[Wes91] WESTOVER L.: *Splatting: A Parallel, Feed-Forward Volume Rendering Algorithm.* PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill, 1991.

[WNDS99] WOO M., NEIDER J., DAVIS T., SHREINER D.: *OpenGL Programming Guide Third Edition.* Addison-Wesley Publishing Ltd, 1999.

[XC04] XUE D., CRAWFIS R.: Efficient splatting using modern graphics hardware. *Graphics Tools 3*, 8 (2004), 1ąV21.

[YZNC05] YUAN X., ZHANG N., NGUYEN M. X., CHEN B.: Volume cutout. *The Visual Computer (Special Issue of Pacific Graphics 2005) 21*, 8-10 (2005), 745–754.

a)   b)   c)   d)
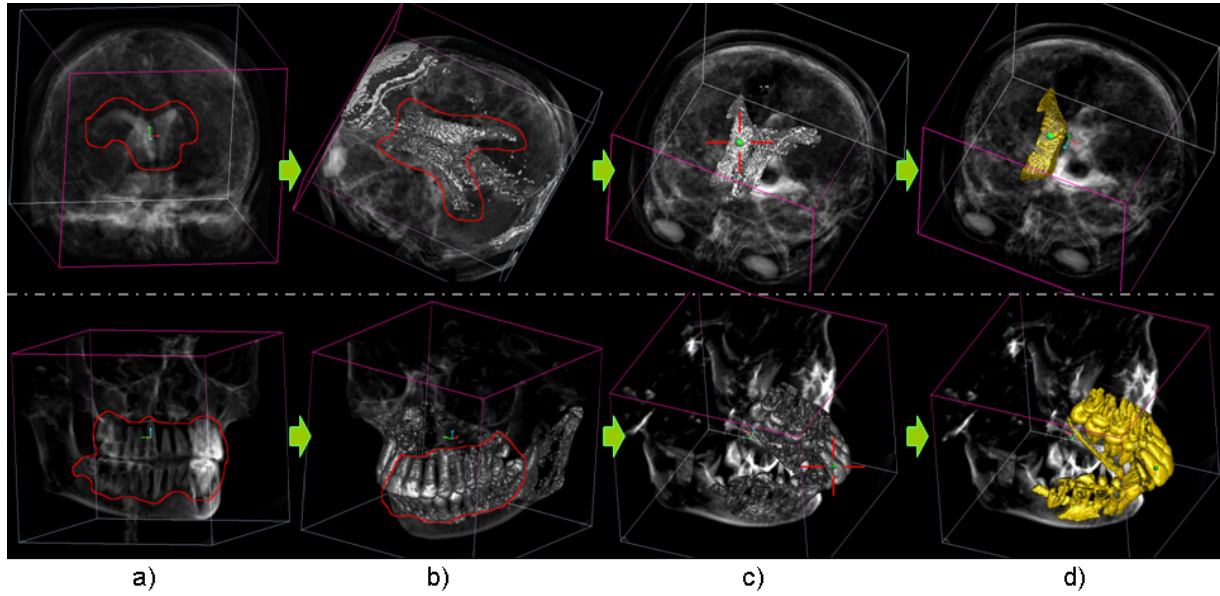
Figure 10: *Segmentation of right ventricle (top) and partial teeth (bottom). (a) Raw volume, X-ray, with sketched-region. (b) Resulting cut, rotating the view, new sketch. (c) Resulting cut, rotating the view, plating the seed. (d) Region growing contained within sketched/resulting volume from (c).*

a)   b)   c)   d)
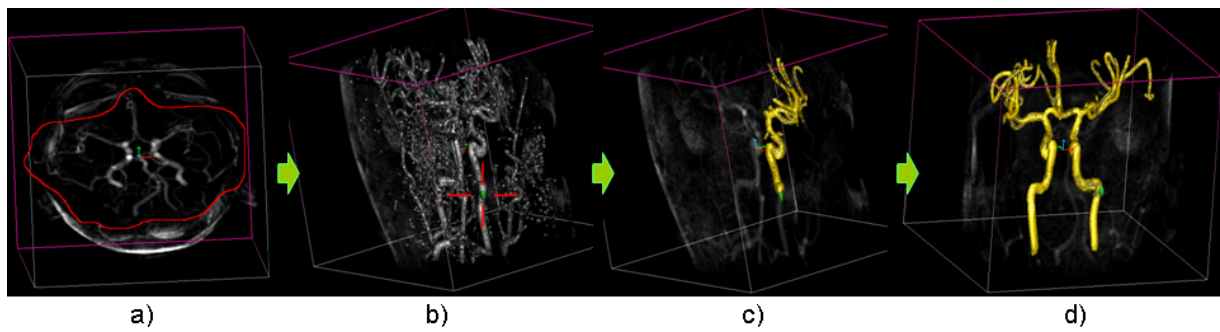
Figure 11: *Segmentation of carotid and cerebral arteries. (a) Raw volume, X-ray, with sketched-region. (b) Resulting cut, rotating the view, plating the seed on the arterial branch. (c,d) Region growing contained within sketched/resulting volume from (b).*

Illustrative Deformation of Volume Data

Carlos D. Correa

Computer Science Department, University of California, Davis

1 Introduction

The purpose of visualization is to gain understanding of 3D structures through images. Although many rendering techniques have been proposed for this purpose, the effective visualization remains a challenging task, due to occlusion, clutter and noise. Solutions to this problem include rendering techniques, abstraction and interaction. Here, we explore a different approach, where the user can manipulate the data set directly. We call this type of manipulation *Illustrative Deformation*. The term *illustrative* can be understood in two ways. One, in the sense that the deformation is inspired by the types of operations that are often depicted in scientific illustration, such as peels and incisions, which are used to improve visibility of features or to better depict a procedure. But also, since the purpose of deformation is to improve visualization, it does not need to be physically correct, but it can be defined empirically. In this tutorial, we will describe a generic pipeline for obtaining such deformations on volumetric models at interactive rates.

2 Rendering Deformation

Deformation refers to the change in time of the position and orientation properties of graphical elements. There has been a considerable amount of research of deformation of surface meshes, where deformation is obtained by directly transforming the vertices of the mesh. Volumetric data sets, in contrast, are represented using voxels, three-dimensional points that include appearance properties, such as opacity and color. Unlike surface meshes, connectivity information is not explicit, which complicated the creation of meaningful deformations. Methods for volume deformation, often coupled with the rendering process, include ray deflectors [14], *free-form deformation* [2], pre-defined point-wise deformation [16], splitting operations [13], and chain-mail algorithms [10]. These methods can be considered as empirical, as opposed to physically based methods. Early approaches addressed deformation at the modeling stage [12, 11, 15, 8, 9]. Due to the large size of volumes and the sampling considerations, it is more practical to couple deformation and rendering in a single stage. For a more complete survey, refer to Chen et al.'s [3] and Nealen et al.'s surveys [17].

In general, we can defined deformation as a mapping $T_F : \mathbb{R}^3 \mapsto \mathbb{R}^3$, such that, for a given point $\mathbf{p}$, we can obtain a new position $\mathbf{p}' = T_F(\mathbf{p})$. We denote T_F as a *forward transformation*. Let P_V be the set of all points $\mathbf{p}$ in the volume representation V of an object. After deformation, we obtain a new set $P_V' = \{\mathbf{p}' | \forall \mathbf{p} \in P_V, \mathbf{p}' = T_F(\mathbf{p})\}$. The new axis-aligned bounding volume for all the points in P_V', denoted as V', is called the *deformed volume*. Forward transformation is of limited use in the deformation of volumetric objects due to the impracticality of using each voxel as a primitive. Instead, volume deformation is commonly implemented using *space warping* techniques. We distinguish two different types of methods: *indirect* and *direct space warping*. At the core of these two is the idea that volume rendering is obtained by sampling the deformed volume V'.

Indirect Space Warping. Indirect or proxy-based space warping is obtained by defining a set of control points which are deformed using a forward transformation. The set of deformed control points are then used to "reconstruct" the embedded volume in the new configuration, usually via interpolation. This is depicted in Figure 1(a). Methods in this category include free-form deformation [24], direct deformation of trilinear patches [19] and skeleton-based deformation techniques [9, 20, 21]. Extending these methods to include cuts is more difficult, as the proxy geometry needs to be tessellated [14].

Direct Space Warping. To avoid dealing with complex tessellation, deformation can be defined as a point-wise warping of the volume. In this case, we need and inverse transformation T_F^{-1} so that, for each point $\mathbf{p}' \in P_V'$ we obtain $\mathbf{p} = T_F^{-1}(\mathbf{p}')$. This approach was been applied to ray tracing systems in the form of ray deflectors [14] and spatial transfer functions (STF) [4]. Direct space warping techniques extend easily to model cuts, by *tagging* points that do not contribute to the final image. Let $P_{V'}'$ be the collection of all points in V'. Since P_V' is a set of all points located in V' with a pre-image in V, the empty space in V' is thus defined by a set of points $P_{empty}' = P_{V'}' - P_V'$. Instead of using the inverse transformation, we warp each point with a *backward transformation* T_B, defined as:

$$\mathbf{p} = T_B(\mathbf{p}') = \begin{cases} T_F^{-1}(\mathbf{p}') & \mathbf{p}' \in P_V' \\ \varnothing & \mathbf{p}' \in P_{empty}' \end{cases} \tag{1}$$

where $\varnothing$ denotes a null position, indicating a point that does not have an origin prior to the manipulation. In general, such points are considered empty, or completely transparent. We thereby assume that, for purposes of rendering, $f'(\varnothing) = 0$. This method is depicted in Figure 3(a). .
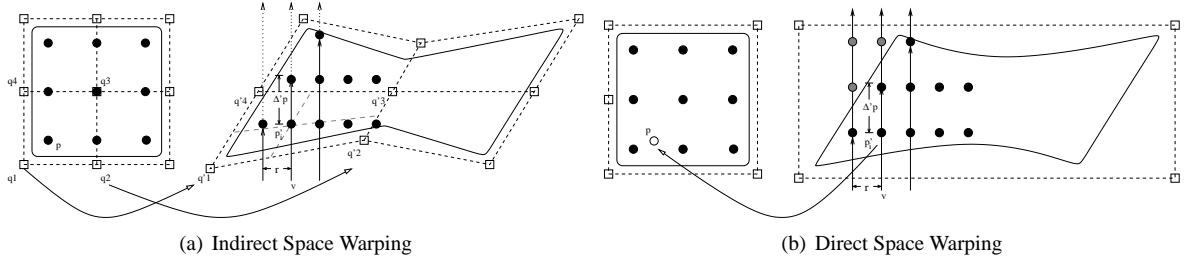
(a) Indirect Space Warping (b) Direct Space Warping

Figure 1: Space warping techniques for Volume Deformation

3   Illustrative Deformation Pipeline

The purpose of this tutorial is to develop the foundations for building a generic volume deformation pipeline. Because of the ability to generate high quality rendering, we focus on direct space warping. The key concept for deforming volumetric objects is that deformation is part of the rendering process. The rendering process, as depicted in Figure 2, is composed of the following stages: (1) *displacement setup*, usually as a pre-process, where deformation templates are defined, (2) Volume *slicing*, where the volume is divided into view-aligned slices (3) for each pixel generated by slicing, we perform *warping* and *sampling* to obtain the deformed sample values, (4) *lighting* and finally (5) *compositing*.

Displacement Setup. Before applying a deformation, we define a series of *templates*, inspired by surgical tools and manipulation operators, which are encoded as *displacement maps*. Example operations are peeling, bending and cutting. Displacement maps are textures that define a spatial displacement rather than a color attribute, and are widely used to add detail to surface models [5, 18, 22]. Here, we introduce a generalized notion of a displacement map, which allows for unconventional features such as unorthogonal and discontinuous displacements [7]. To create a displacement texture, D we first specify the forward operation $\overrightarrow{D}^{\triangleright}$ procedurally, and then sample its inverse transformation $D = \overrightarrow{D}^{\triangleleft}$ at discrete positions. Because of the presence of cuts, the inverse may not be defined for all points in the domain of the deformation, so D is extended such that at least C^0 continuity is obtained. This is done by creating an alpha mask, where a value of 1 means that the point has a pre-image in the co-domain of $\overrightarrow{D}^{\triangleright}$ and 0 otherwise. To avoid aliasing artifacts due to the binary mask, we define the final alpha mask A as its distance field.

Warping and Sampling. In order to determine the displaced volume, we slice the proxy scene geometry into view-oriented slices. The bounding box of V' can easily be found by combining the bounding boxes of the object(s) and their displacements. For every pixel generated, the *warped coordinate* is then computed as:

$$\mathbf{p} = T_F^{-1}(\mathbf{p}') = \mathbf{p}' + D(\mathbf{p}') \tag{2}$$

Next, we sample the scalar field f at the position $\mathbf{p}$ and retrieve the volume values. Finally, in order to handle discontinuities, we sample the alpha mask A at the position $\mathbf{p}'$ and modulate the pixel's color components with the mask, as follows:

$$f'(\mathbf{p}') = \begin{cases} f(\mathbf{p}) & A(\mathbf{p}) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The use of a smooth map to define continuity ensures that the surfaces of cuts are rendered with high-quality and without aliasing artifacts.

Displaced Surface Normal. In order to properly shade the object, we need the normal information at each point. Since we store objects as volumes, normals can be obtained using finite differences or can be pre-computed and stored in a 3D texture. Because of deformations, normals need to be computed in *deformed* space, or transformed from the original normals. Because of speed and quality, it is better to transform the normals. The new normal at $\mathbf{p}'$ can be obtained by transforming the original one at $\mathbf{p}$ using the Jacobian of the deformation, as proposed by Barr [1]. The normal transformation is as follows:

$$\overrightarrow{\mathbf{n}}^{(p')} = (\mathsf{I} + \mathsf{J}_D^{(p')})^{\top} \overrightarrow{\mathbf{n}}^{(p)} \tag{4}$$

where $\overrightarrow{\mathbf{n}}^{(p)}$ is the original undeformed normal, J_D is the Jacobian of the displacement field and I is the identity matrix. Another type of normal is the one introduced by a cut, since new surfaces may appear. To adjust the normal at the surface without introducing aliasing artifacts, we gradually correct the normal in the vicinity of the cut to the desired normal, via blending:

$$\overrightarrow{\mathbf{n}}^{(p')} = \omega(\mathsf{I} + \mathsf{J}_D)^{\top} \overrightarrow{\mathbf{n}}^{(p)} + (1-\omega)\nabla_A^{(p')} \tag{5}$$

where $\omega \in [0,1]$ is a blending factor. This blending mechanism is similar to the solution proposed by Weiskopf et al. [23] for volumetric cutaways.
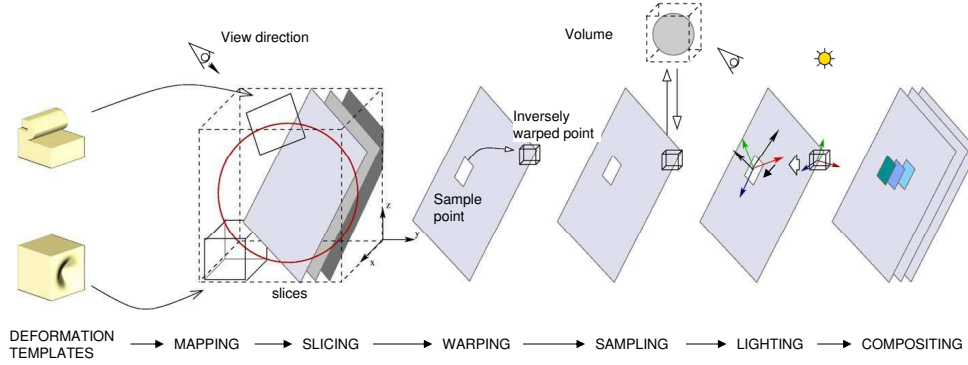
Figure 2: Overview of Illustrative Deformation of volumes

3.1 Overview of the algorithm

In summary, the algorithm for illustrative deformation is as follows:

Algorithm 3.1: DEFORMATION ALGORITHM(...)

COMPUTE DEFORMED BOUNDING VOLUME(B)
for each *sample* $\mathbf{p}' \in B$

do
$\begin{cases}
\mathbf{p} \leftarrow \mathbf{p}' + D(\mathbf{p}') \\
s \leftarrow V(\mathbf{p}) \\
\alpha_D \leftarrow A(\mathbf{p}') \\
n \leftarrow \text{ESTIMATE ORIGINAL GRADIENT}(\mathbf{p}') \\
n' \leftarrow \text{TRANSFORM NORMAL}(n) \\
c, \alpha \leftarrow \text{CLASSIFY AND LIGHT}(s, n') \\
\textbf{if } \alpha_D \leq 0 \\
\quad \textbf{then } \alpha = 0 \\
\text{COMPOSITE}(c, \alpha)
\end{cases}$

The function call *Estimate Original Gradient* can be implemented using finite differences, or sampling from a pre-computed gradient volume. The function call *Transform Normal* applies the operation in Eq.(4), and the function call *Classify and Light* maps density values to color and opacity, based on transfer functions and a local illumination model. This algorithm can be easily implemented on a GPU-based volume renderer, as part of the pixel shader. For a texture-based renderer, the bounding volume B is sliced in a view-aligned manner. For a GPU raycaster, each pixel generates a ray, which is sampled along the view direction to generate the points $\mathbf{p}'$.

3.2 Interaction with Displacement Maps

An important aspect of this approach, is the ability to deform volumes at interactive rates. Displacement maps can be thought of as warping objects that can be moved, rotated or scaled arbitrarily within the target volume. For example, by translating a peeling tool, the user can interactively open and close a peeled region from a volume. By scaling a cutting operation, the user can increase or decrease the size of the incision.

We can generalize this notion by allowing affine transformations to be performed on the sample points before deformation, i.e. as an extra warping operation. We can extend the displacement equation to:

$$\hat{\mathbf{p}} = \mathsf{M} \times \left(\mathsf{M}^{-1}\hat{\mathbf{p}}' + D(\mathsf{M}^{-1}\hat{\mathbf{p}}') \right) \tag{6}$$

where M is a 4×4 affine transformation and the points are given in homogeneous coordinates. The normal transformation when undergoing this type of coordinate transformation is given by the concatenation of the transpose of the Jacobians of the coordinate transformation [7], which leads to:

$$\overrightarrow{\mathbf{n}}^{(p')} = \left[(\mathsf{S}^{-1} \times \mathsf{R})\,(\mathsf{I} + \mathsf{J}_D)^{\top}\,(\mathsf{R}^{\top} \times \mathsf{S}) \right] \overrightarrow{\mathbf{n}}^{(p)} \tag{7}$$

Where the affine transformations is represented as a rotation (R) followed by a scaling (S) and a translation (T) , i.e., $\mathsf{M} = \mathsf{T} \times \mathsf{S} \times \mathsf{R}$.

(a) Cuts for axis-aligned deformation   (b) Cuts with mask-based deformation
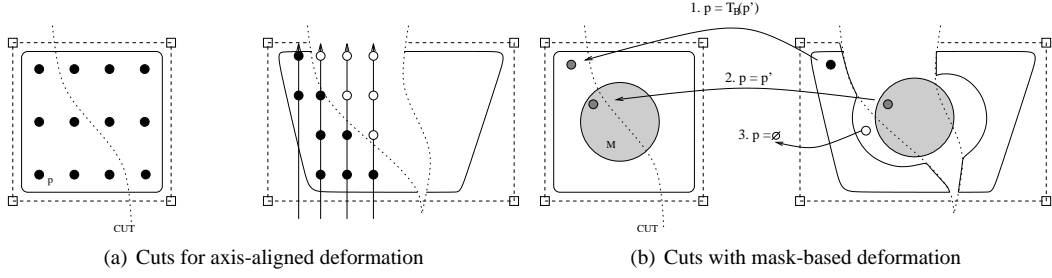
Figure 3: Simulation of cuts with direct space warping

This transformation via affine matrix is an efficient mechanism for controlling the shape, position and orientation of a displacement map. Since this method only involves multiplication of constant matrices, this is an efficient alternative for controlling the displacement in an interactive application.

4 Feature Aligned Deformation

Because the above algorithm works directly on the volume data, it is insensitive to features of interest. In many cases, it is desired to *align* an operator so that it follows a surface of interest. But since objects of interest can be complex, a geometric transformation of the deformation may be difficult to derive. Instead, a *masking* operator yields acceptable results. The key idea is to modulate the deformation so that certain tagged points do not undergo deformation. In order to do that, we introduce a *masking function M*, which defines the feature-sensitivity of points in the original volume V, and is typically represented by a volume data set. When $M(\mathbf{p}) < 0$, $\mathbf{p}$ is part of the feature to be preserved, and cannot be transformed. Let P_V be the set of points of the original volume and P'_V the deformed set of points. Let P_M be the subset of P_V, such that $P_M = \{\mathbf{p} | \mathbf{p} \in P_V, M(\mathbf{p}) < 0\}$, and V_M is an axis-aligned bounding volume of P_M. Any point not in P_M is operatable [6].

Modified Warping. In feature-aligned deformation, an inversely transformed point $\mathbf{p}'$ may have been masked as non-operatable by M, which results in empty space. To handle the complexity of this inverse mapping, we introduce an initial "probe" $\mathbf{p}^0 = T_B(\mathbf{p}')$. We then obtain the warped position $\mathbf{p}$ by taking the feature mask into account as:

$$\mathbf{p} = \begin{cases} \mathbf{p}^0 & \mathbf{p}^0 \in P_V \wedge (M(\mathbf{p}') \geq 0 \wedge M(\mathbf{p}^0) \geq 0) \\ \mathbf{p}' & \mathbf{p}' \in P_V \wedge M(\mathbf{p}') < 0 \\ \varnothing & \text{otherwise} \end{cases} \tag{8}$$

These three cases are shown in Figure 3(b), namely: (1) the point is transformed, (2) the point is masked and therefore untransformed, and (3) the point is empty due to the feature-aligned cut. One example definition of a mask is a distance field of a feature of interest, such that it is positive in the interior of the object and negative on the outside. Points lying in the surface of the feature have mask value zero.

Modified Normal Estimation. Similar to the original algorithm, the normals must be transformed. However, the normal at a point is influenced by an additional factor, which is the case for those points in the vicinity of the surface of the feature of interest. This can be done in two consecutive blending operations. The first blending is defined in Eq.(5), which adjusts the normal near cuts. The second blending adjusts the normal near the surface of the feature of interest. The final normal at a point is then a combination of this adjusted normal and the normal of the surface ∇_M.

$$\overrightarrow{\mathbf{n}}^{(p')} = \beta_1 \overrightarrow{\mathbf{n}}_T^* + \beta_2 \nabla_M \tag{9}$$

where the weighting factors β_1 and β_2 are chosen so that the normal for a point in the surface of an object is ∇_M, and for a point in the underside of the volume after removing the feature (when undergoing deformation) is $-\nabla_M$.

4

(a) CT Head peel   (b) Forefoot incision

Figure 4: Comparative Results of Illustrative Deformation Methods

4.1 Overview of the modified algorithm

We now modify the original deformation algorithm, to account for the mask defining features of interest.

Algorithm 4.1: MODIFIED DEFORMATION ALGORITHM(...)

COMPUTE DEFORMED BOUNDING VOLUME(B)
for each *sample* $\mathbf{p}' \in B$
do $\begin{cases} \mathbf{p} \leftarrow \mathbf{p}' + D(\mathbf{p}') \\ m \leftarrow M(\mathbf{p}') \\ \alpha_D \leftarrow A(\mathbf{p}') \\ \textbf{if } m \geq 0 \\ \quad \textbf{then } \begin{cases} m \leftarrow M(\mathbf{p}) \\ \textbf{if } m < 0 \\ \quad \textbf{then } \alpha_D \leftarrow 0 \end{cases} \\ \quad \textbf{else } \begin{cases} \mathbf{p} \leftarrow \mathbf{p}' \\ \alpha_D \leftarrow 1 \end{cases} \\ s \leftarrow V(\mathbf{p}) \\ n \leftarrow \text{ESTIMATE ORIGINAL GRADIENT}(\mathbf{p}') \\ n0 \leftarrow \text{TRANSFORM NORMAL}(n) \\ n' \leftarrow \text{ADJUST NORMAL}(n0, m) \\ c, \alpha \leftarrow \text{CLASSIFY AND LIGHT}(s, n') \\ \textbf{if } \alpha_D \leq 0 \\ \quad \textbf{then } \alpha = 0 \\ \text{COMPOSITE}(c, \alpha) \end{cases}$

The main difference with the original algorithm is the explicit handling of cases for finding the proper deformed point, as depicted in Figure 3(b). In addition, a new function call *Adjust Normal* is required to apply the extra transformations shown in Eq.(9).

5 Example Illustrations

Figure 4 shows a comparative table of applying mask-based deformations to two CT data sets, showing the original volume, the axis-aligned deformation, and feature-aligned deformation. On the left, we see a peeler deformation applied to the CT head data set, and an incision on a forefoot CT data set on the right. Figure 5(a) shows an application of a continuous deformation. In this case, we simulate an illustration of a whiplash action, with a bending operation. Figure 5(b) shows an illustration of one stage of a craniotomy. We applied this to the CTHead data set, after an approximate segmentation of the skull, and also to a portion of the segmented Visible Man data set. Figure 5(c) illustrates two stages in a frog dissection procedure, where the user can interactively control the size and depth of the incision tool.

6 Summary

This part of the tutorial described a general pipeline to obtain illustrative deformations. This type of manipulations are often found in surgical illustration, where the depiction of cuts and deformation help understand a procedure, provide context, or elucidate the shape of an object. We showed a practical specification of deformations using *displacement maps*, which can be implemented efficiently in contemporary programmable hardware using 3D textures. To obtain high

(a) Illustration of whiplash injury (b) Illustration of a craniotomy (c) Virtual frog dissection
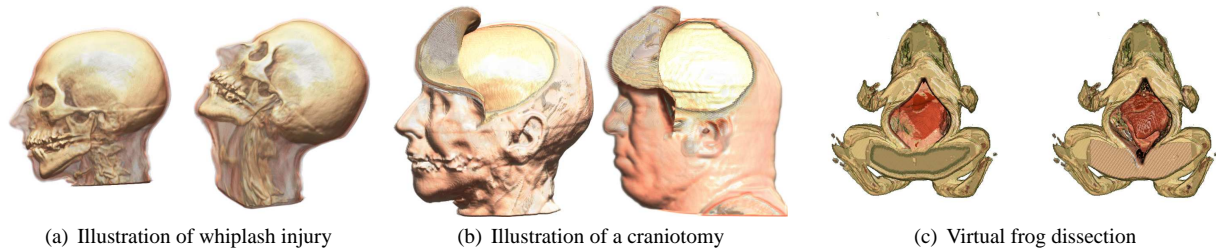
Figure 5: Example Illustrative Deformations

quality rendering of deformed volumes, we make use of *mask* volumes to (1) control the shape of cuts to avoid aliasing artifacts and (2) preserve features of interest. We also showed a series of transformations to adjust the normals so that the isosurfaces of interest are correctly depicted and lit according to a local illumination model. Through a number of examples, we have shown the flexibility and operatability of these methods, and how illustrative deformation can be achieved at interactive rates.

References

[1] Alan H. Barr. Global and local deformations of solid primitives. In *SIGGRAPH '84: Proc. of the 11th annual conference on Computer graphics and interactive techniques*, pages 21–30, New York, NY, USA, 1984. ACM Press.

[2] H. Chen, J. Hesser, and R. Manner. Ray casting free-form deformed-volume objects. *The Journal of Visualization and Computer Animation*, 14:61–72(12), 2003.

[3] M. Chen, C. Correa, S. Islam, W. Jones, P.Y. Shen, D. Silver, S. J. Walton, and J. Willis. Manipulating, deforming and animating sampled object representations. *Computer Graphics Forum*, 26(4):824–852, December 2007. 0167-7055.

[4] M. Chen, D. Silver, A. S. Winter, V. Singh, and N. Cornea. Spatial transfer functions: a unified approach to specifying deformation in volume modeling and animation. In *Proc. Volume Graphics '03*, pages 35–44. ACM Press, 2003.

[5] R. L. Cook. Shade trees. *Computer Graphics (Proc. SIGGRAPH 84)*, 18(3):223–231, 1984.

[6] Carlos Correa, Deborah Silver, and Min Chen. Feature aligned volume manipulation for illustration and visualization. *IEEE Trans. on Visualization and Computer Graphics (Proc. Visualization / Information Visualization 2006)*, 12(5):1069–1076, September-October 2006.

[7] Carlos D. Correa, Deborah Silver, and Min Chen. Discontinuous displacement mapping for volume graphics. In *Proceedings of the Fifth Eurographics / IEEE VGTC Workshop on Volume Graphics 2006*, pages 9–16. Eurographics Association, 2006.

[8] S. Fang, R. Raghavan, and J. T. Richtsmeier. Volume morphing methods for landmark-based 3D image deformation. In M. H. Loew and K. M. Hanson, editors, *Proc. SPIE Vol. 2710, p. 404-415, Medical Imaging 1996: Image Processing, Murray H. Loew; Kenneth M. Hanson; Eds.*, pages 404–415, April 1996.

[9] Nikhil Gagvani and Deborah Silver. Animating volumetric models. *Graph. Models*, 63(6):443–458, 2001.

[10] Sarah F. Gibson. 3d chainmail: a fast algorithm for deforming volumetric objects. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 149–ff., New York, NY, USA, 1997. ACM Press.

[11] Taosong He, Sidney Wang, and Arie Kaufman. Wavelet-based volume morphing. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 85–92, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[12] John F. Hughes. Scheduled fourier volume morphing. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 43–46, New York, NY, USA, 1992. ACM Press.

[13] Shoukat Islam, Deborah Silver, and Min Chen. Volume splitting and its applications. *IEEE Trans. on Visualization and Computer Graphics*, 13(2):193–203, 2007.

[14] Yair Kurzion and Roni Yagel. Interactive space deformation with hardware-assisted rendering. *IEEE Comput. Graph. Appl.*, 17(5):66–77, 1997.

[15] Apostolos Lerios, Chase D. Garfinkle, and Marc Levoy. Feature -based volume metamorphosis. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 449–456, New York, NY, USA, 1995. ACM Press.

[16] Michael J. McGuffin, Liviu Tancau, and Ravin Balakrishnan. Using deformations for browsing volumetric data. In *Proceedings of IEEE Visualization (VIS) 2003*, pages 401–408, October 2003.

[17] A. Nealen, M. Muller, R. Keiser, E. Boxerman, and M.Carlson. Physically based deformable models in computer graphics. In *Eurographics STAR Report*, 2005.

[18] M. Pharr and P. Hanrahan. Geometry caching for ray-tracing displacement maps. In *Proc. Eurographics Rendering Workshop*, pages 31–40, 1996.

[19] C. Rezk-Salama, M. Scheuering, G. Soza, and G. Greiner. Fast volumetric deformation on general purpose hardware. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 17–24, New York, NY, USA, 2001. ACM Press.

[20] V. Singh, D. Silver, and N. Cornea. Real-time volume manipulation. In *Proc. Volume Graphics '03*, pages 45–51. ACM Press, 2003.

[21] S.J. Walton and M.W. Jones. Volume wires : A framework for empirical non-linear deformation of volumetric datasets. *Journal of WSCG 2006*, 14, 2006.

[22] L. Wang, X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum. View-dependent displacement mapping. *ACM Trans. on Graphics (Proc. SIGGRAPH 2003)*, 22(3):334–339, 2003.

[23] Daniel Weiskopf, Klaus Engel, and Thomas Ertl. Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Trans. Vis. Comput. Graph.*, 9(3):298–312, 2003.

[24] Rüdiger Westermann and Christof Rezk-Salama. Real-time volume deformations. *Comput. Graph. Forum*, 20(3), 2001.

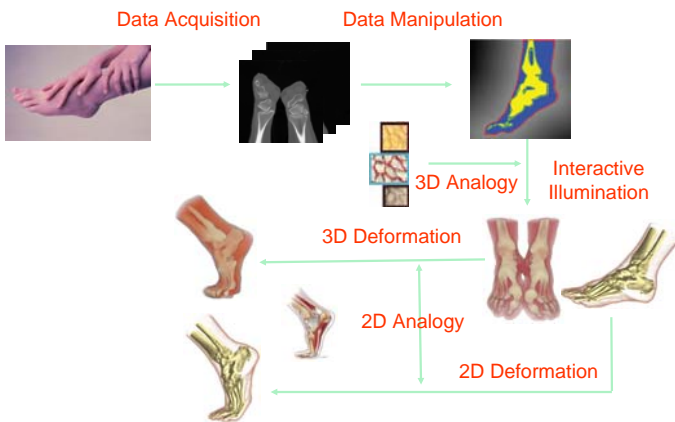# Example-based Illustrative Modeling and Rendering

Wei Chen,
chenwei@cad.zju.edu.cn
Zhejiang University, Purdue University

Motivation

❖ **Learn from examples for case that are difficult to represent and model**
- Existing 2D illustrations
- Existing models and datasets

❖ **Fulfill example-based illustration by means of**
- Shape deformation
- Texture synthesis

Computer-generated illustration

Data Acquisition   Data Manipulation
3D Analogy   Interactive Illumination
3D Deformation   2D Analogy   2D Deformation

Overview

❖ **Shape and shape variations by examples**
- Convey objects from measured datasets
- Interactive shape manipulation
- Example-based shape transfer

❖ **Appearance and rendering styles by examples**
- Texture synthesis and transfer
- Rendering styles by examples

Modeling from measured data

❖ **Boundary shapes**
- Iso-surface [Lorrence87]
- Volumetric image processing [Whitaker00]
- Transfer function-based [Kitware]

Modeling from measured data

❖ **Structural information**
- CT [Dong05]
- DTI [Wenger04]

DTI data

Visible Human   Modeled
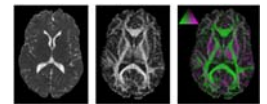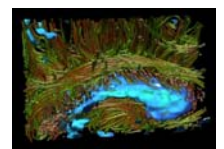
Fiber illustration

Modeling from measured data

❖ **Texture and appearance**
  - Vision [Dorsey04]
  - Capture [Gross07]
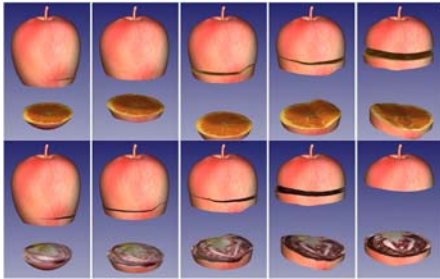
Interactive manipulation

❖ **Volume deformation**
  - See Carlos D. Correa

(a)   (b) *Illustration*   (c)   (d) *Illustration*
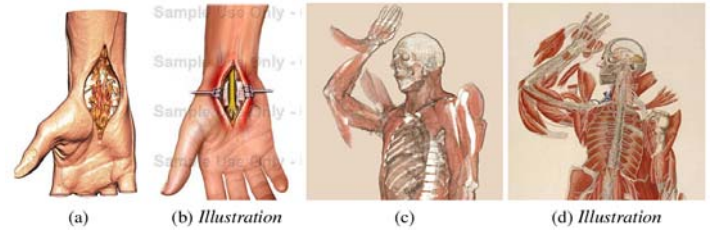
Interactive manipulation

❖ **Surface deformation**
  - Freeform deformation
  - Skeleton deformation
  - Mesh deformation

[Funck06]
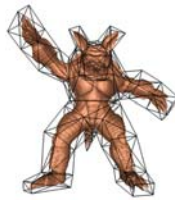
[Sederberg86]   [Ju06]   [Yu05]

Example based shape transfer

❖ **3D to 3D [Sorkine04, Yu04]**
  - Transfer locally encoded details

$$\delta_i = \frac{1}{d_i} \sum_{v \in N(i)} (\mathbf{v}_i - \mathbf{v})$$
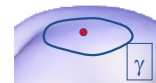
$$\frac{1}{len(\gamma)} \int_{v \in \gamma} (\mathbf{v}_i - \mathbf{v})\, ds$$
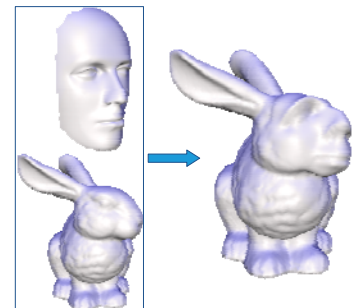
Example based shape transfer

❖ **2D to 3D [Zelink04]**
  - Using curves to modify surface contour
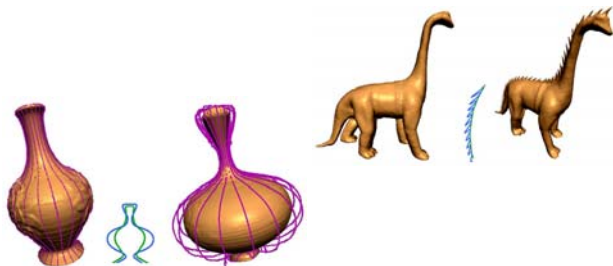
Example based shape transfer

❖ **2D to 3D [Zhou06]**
  - Using curves to drive deformation
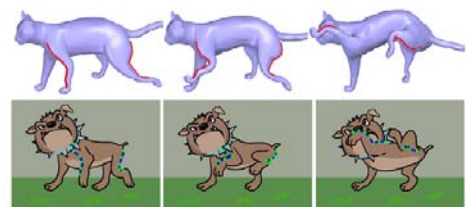
Example based shape transfer

❖ **2D to 3D [Chen07]**
  ▪ Using curve to drive deformation

Live demo
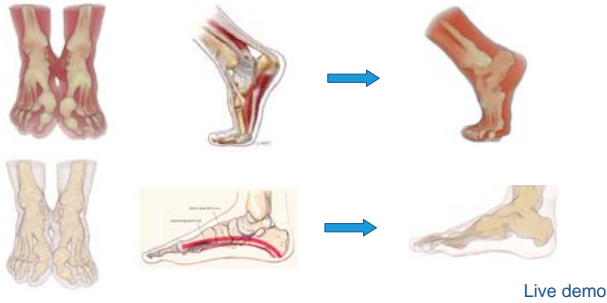
Example based shape transfer

❖ **2D to 3D [Chen07]**
  ▪ Using distance field to convert surface to data to get smooth boundary effects

Example based shape transfer

❖ **3D stylization from 2D example**
  ▪ Context curves, silhouette, feature points, local geometric details
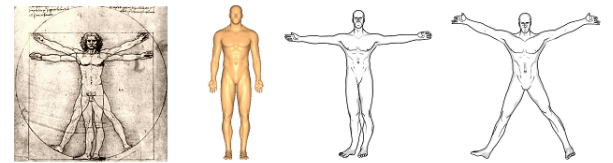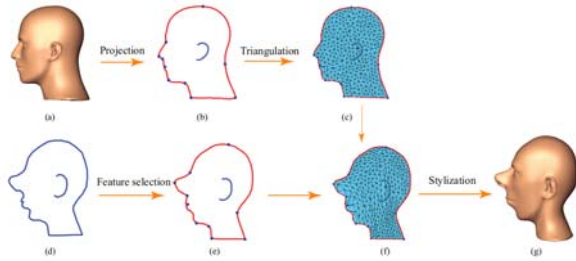
Video demo

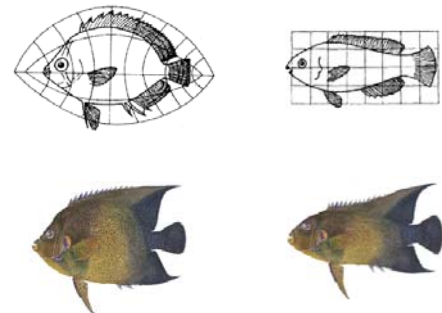## Example based shape transfer

❖ **2D deformation by example**
  ▪ Differential based 2D mesh manipulation
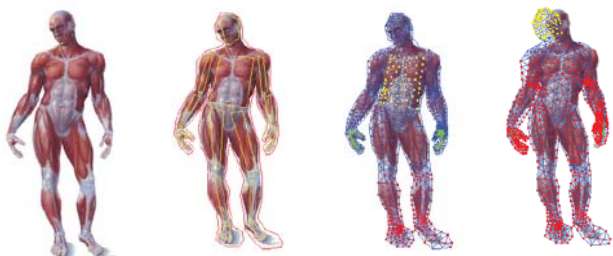
Example based shape transfer

❖ **2D deformation by example**
  ▪ Example-based shape manipulation

Example based shape transfer

❖ 2D deformation by examples

- Flexible post-process to modify the results
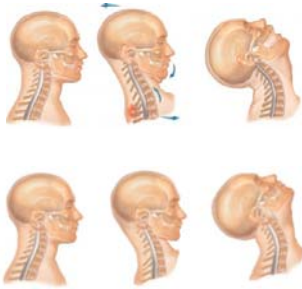
Video demo

Example based deformation transfer

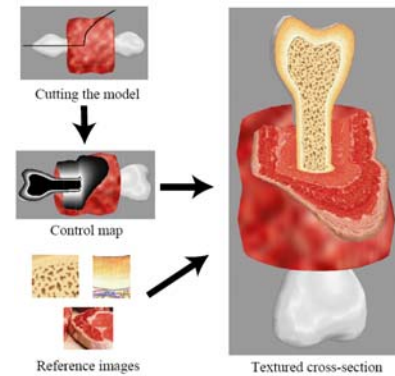## ❖ 3D to 3D [Sumner04]
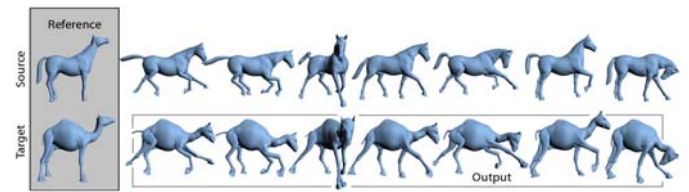
Example-based appearance transfer

❖ Color transfer [Lu05]
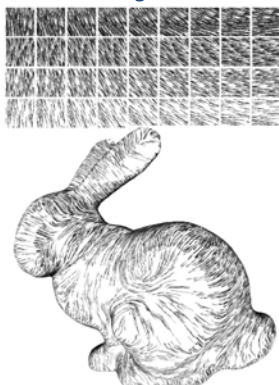
- Simple representation and similar distribution
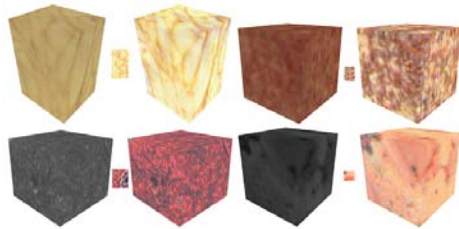
Example-based appearance transfer

❖ 2D texture synthesis [Owada04]

Example-based appearance transfer

❖ Surface texture synthesis [Gorla03]

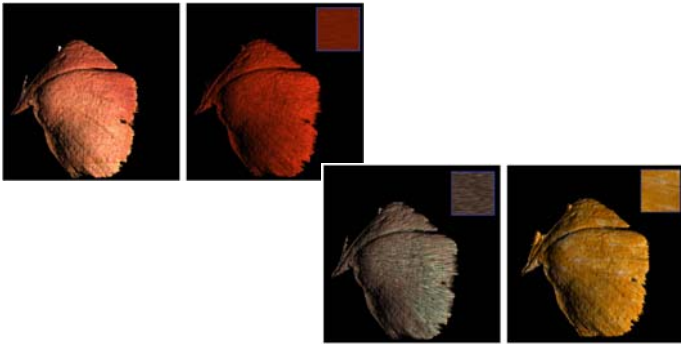# Example-based appearance transfer

❖ Solid texture synthesis [Lu05]

- Simulate styles of professional illustrators
- Simplify user interaction

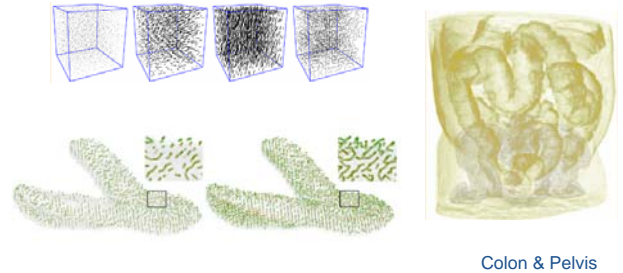## Example-based appearance transfer

❖ **Solid texture synthesis [Dong05]**
- Synthesize the texture guided by the vector field from visible human

Example-based appearance transfer

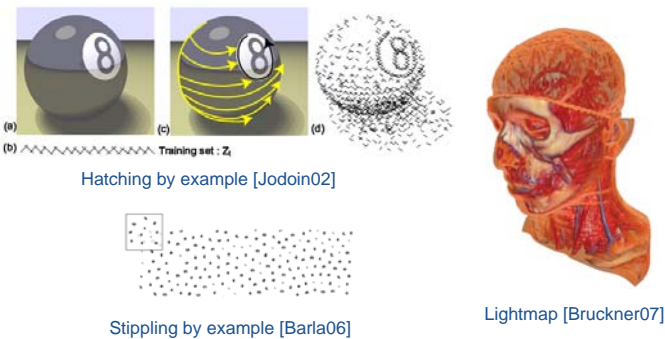❖ **Solid texture synthesis [Lu07]**
- Wang cube for non-periodic patterns

Colon & Pelvis

Example-based appearance transfer

❖ **Rendering styles transfer**

Hatching by example [Jodoin02]

Stippling by example [Barla06]

Lightmap [Bruckner07]

Conclusions

❖ **Transfer intrinsic features from multiple sources**

❖ **Employ multiple styles in illustrations**

❖ **Always keep the user in the interaction loop**

Acknowledgments

❖ **Image courtesy:**
- VTK, Feng Dong, Keipfer Wenger, Julie Dorsey, Marcus Gross, Carlos Correa, Thomas Sederberg, Tao Ju, Yizhou Yu, Wolfram von Funck, Robert Sumner, Orga Sorkine, Kun Zhou, Pierre-Marc Jodoin, Shigeru Owada, Aidong Lu, Pascal Barla, Gabriele Gorla, Nikolai Svakhine, Stefan Bruckner

❖ **Collaborators:**
- David S.Ebert, Aidong Lu, Song Zhang, Guanghua Tan, Ligang Liu, Nikolai Svakhine, Xiao Liang, Stephen Correia

Thank You !

chen23@purdue.edu
http://web.ics.purdue.edu/~chen23