Palpable Visualizations: Techniques for Creatively Designing Discernible and Accessible Visualizations Grounded in the Physical World

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Seth Alan Johnson

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy of Science

Daniel F. Keefe

July, 2020

© Seth Alan Johnson 2020
ALL RIGHTS RESERVED

Acknowledgements

I would like to express my gratitude to the many people who have supported, encouraged, and guided me through the work leading to this dissertation. Without all of you, I neither would nor could have undertaken this endeavor.

A particular joy of researching interdisciplinary collaboration techniques is the relationships with the interdisciplinary collaborators. Without the teamwork of colleagues and co-authors such as these, none of the works in this dissertation would exist. First, I am grateful for the partnership of the staff and researchers at the Earl E. Bakken Medical Devices Center, particularly Arthur Erdman and Bethany Juhnke. I am also thankful to the team of architects from MINN_LAB, Ross Altheimer, Robert Hunter, Andrea Johnson, Maura Rockcastle, Marc Swackhamer, Aaron Wittkamper, and Deuk-Geun Hong for welcoming me into their creative design process for a year and sharing the most crowded and rainy all-nighter of my academic career. And I am humbled by the patience and enthusiasm of the domain scientists who have partnered with us to find new means of exploring their data; in particular, Ashish Singal, Marcos Molina, Hakizumwami Birali Runesha, Lingyu Meng, Phil Wolfram, and Christophe Lenglet.

Likewise, I am delighted to have spent the last three years working closely with the students and staff of The University of Texas at Austin and the Texas Advanced Computing Center, including Andrew Solis, Hannah Simon, Stephanie Zeller, Annie Bares, and the countless others who were always welcoming when I visited. And to Greg Abram, thank you for sharing your wisdom, your lunch breaks, and your homes with me; working with you has been a pleasure and an honor.

I have been especially inspired and encouraged by each of my committee members. Thank you to Evan Suma Rosenberg and Stephen Guy, for pouring fresh fuel into my passion for virtual reality and computer graphics through your research and your

teaching, and for understanding and supporting my career goals as they evolved. And Francesca Samsel, thank you for your creativity, your chocolate, and your championing of everything we built together. I wouldn't trade our wild adventure for anything; never stop finding new problems to solve.

I can only begin to express my gratitude for my advisor, Dan Keefe. From inviting me into research, to patiently working with me to discern my calling, you've given me every opportunity to succeed— and together we discovered what success means for me. Through your mentorship and compassion for not only me but each of my fellow students, you have taught me at least as much about grace and leadership as you have about the study of 3D user interfaces. Your commitment to fostering diversity and character in your lab has been a blessing to me daily, and for every new student you welcome into the family I am only further humbled to be a part of it.

To my fellow Ph.D. lab mates Jung, Dan O, Bridger, and Volcano, I can't imagine the last six years without having shared deadlines, lunches, stories, hotels, jokes, and adventures with coworkers who were even more so my friends. To Nikki, Liam, Daniel, Devin, and all who made 2-203 their home, each of you made coming to work worthwhile on the hardest days. And to Zach Fuenning, David Cook, Michelle Kleckler, Matt Overby, John Harwell, Moses Adeagbo, Sarah McRoberts, Michael Tetzlaff, Amy Nippert, and all the rest of my friends who started their doctorates, thanks for your encouragement, honesty, and solidarity as we struggled together — I'm impressed by each and every one of you no matter where your paths have taken you.

And I cannot overstate the gift of everyone else who helped my life be about more than simply school. Every member of my church small group, my incredible housemates, my D&D and board-gaming groups, GCF book club, the U-Swing club, rock climbing buddies, Campus Outreach, and all the rest of the friends I've made during and before this program, each of you reminded me that I was never putting my life on hold.

And finally, my gratitude is owed to the people who sowed the seeds that brought me to this path: Peter Melling, for asking me if I knew what computer science was and mentoring me while I learned; my Normandale instructors Kevin Lee, Sharon Harvey, and Jim Polzin, who taught me the puzzle-solving super-power of programming and urged me onward; and to my parents, who raised me to love learning, explore my passions, and to know my identity is rooted in something greater than academic success.

For John Carlis
I wish you could have seen my (nearly) one-draft dissertation;
I couldn't have done it without you.

Abstract

This dissertation investigates techniques to leverage creative processes like sketching, sculpting, and design iteration to improve the discernibility and accessibility of immersive volumetric data visualizations. Discernible visualizations support a viewer's ability to make sense of complexities such as multi-dimensional climate or engineering simulation data. Accessible data visualization both supports the contribution of previously under-utilized design expertise (i.e. artist-accessible visualization design), and subsequently provides access for a broad audience to engage with data through an emphasis on human connection and support for a wide range of displays. Such visualizations aim to provide a palpable, data-driven experience for scientists, artists, and the public.

Three early works are presented as a rationale for investigating Palpable Visualizations. Bento Box, an immersive visualization system for comparing multiple time-varying volumetric simulation ensemble instances, demonstrates a current state-of-the-art for scientific data visualization. Weather Report, an interactive site-specific artwork visualizing six decades of weather data, takes an in-depth look at what can be accomplished when designing data-driven experiences in close collaboration with professional designers. And Lift-Off, a VR-based modeling program designed for artists, shows how creative sketching in both the physical and virtual worlds can result in a more accessible environment for both scientific and design-oriented tasks.

Based on observations from these three prior works, we present Artifact-Based Rendering (ABR), a framework of algorithms and processes that makes it possible to produce real, data-driven 3D scientific visualizations with a visual language derived entirely from colors, lines, textures, and forms created using traditional physical media or found in nature. ABR addresses three current needs: (i) designing better visualizations by making it accessible for non-programmers to rapidly design and critique many alternative data-to-visual mappings; (ii) expanding the visual vocabulary used in scientific visualizations to enable discernment of increasingly complex multivariate data; (iii) bringing a more engaging, natural, and human-relatable handcrafted aesthetic to data visualization to make the resulting data-driven images more accessible and discernible to the viewer.

Finally, we support the accessibility of visualizations through a data streaming and remote rendering pipeline, culminating in demonstrations bridging live supercomputer simulation data with untethered affordable head-mounted AR/VR displays.

Contents

Acknowledgements **i**

Abstract **iv**

List of Figures **ix**

1 Introduction **1**

1.1 Art and Science 2
1.2 Palpable Visualizations 4
1.3 Thesis Statement 5
1.4 Contributions 6
1.5 Overview of this Dissertation 6

2 Related Work **8**

2.1 Making Visualizations Discernible 8
2.2 Creative Design 9
2.3 Grounding Visualizations in the Physical World 11

3 Bento Box: Current State-of-the-Art for Immersive SciVis **13**

3.1 Introduction 14
3.2 Related Work 18
  3.2.1 Ensemble Visualization & Comparative Visualization 18
  3.2.2 Flow Visualization & Animation 20
  3.2.3 Bimanual & 3D User Interfaces 21
3.3 Bento Box: Concept, Visual Layout & Interface 22

3.3.1 Concept and Visual Layout 23
3.3.2 Zooming and Reframing the Widget 24
3.3.3 Creating and Reframing Sub-Volumes 26
3.3.4 Changing the Visualization with the Design Palette 28
3.3.5 Using the Interactive Timeline 29
3.3.6 Rendering Multiple Clipped Volumes 30
3.4 Application and Results 31
3.4.1 Background: Cardiac Leads in the Right Atrium 31
3.4.2 Sampling and Visualizing Solid Domain Data 32
3.4.3 Sampling and Visualizing Fluid Domain Data 34
3.4.4 Expert User Evaluation and User Feedback 36
3.4.5 Memory Usage and Rendering Performance 39
3.5 Discussion of Limitations and Future Work 40
3.6 Conclusion 42

4 Studies in Accessible Design 45
4.0.1 Introduction 46
4.1 Study 1: Weather Report 47
4.1.1 Weather Report Concept 48
4.1.2 Visualization and Interaction Design 51
4.1.3 Observations, Surprises, and Reflections 55
4.2 Study 2: Lift-Off for Medicine 56
4.2.1 Introduction 56
4.2.2 Related Work 58
4.2.3 Application 1: Annotation of Medical Data 59
4.2.4 Application 2: Immersive Medical Device Design 64
4.3 Conclusions 67
4.3.1 Weather Report 67
4.3.2 Lift-Off 69
4.4 Conclusion 70

5 A Theory and Implementation of Artifact-Based Rendering 71
5.1 Introduction 71

5.2 Related Work . . . 76
    5.2.1 Artistic Techniques and Theories in Visualization . . . 76
    5.2.2 Artists and Designers in Visualization . . . 77
    5.2.3 Data Physicalization and Human Connection . . . 77
    5.2.4 Colormaps and Textures for Visualization . . . 78
5.3 Artifact-Based Rendering for Visualization . . . 80
    5.3.1 Stage 1: Creating and Curating Artifacts . . . 81
    5.3.2 Stage 2: Digitizing, and Translating Artifacts . . . 81
    5.3.3 Stage 3: Data-Visual Mapping and Visualization . . . 87
    5.3.4 Data Management . . . 97
5.4 Discussion . . . 100
    5.4.1 ABR Design Guidelines . . . 100
    5.4.2 Pairing with Perceptual Guidelines . . . 102
5.5 Conclusion . . . 102

6 Applications and Results of Artifact-Based Rendering 104

6.1 Introduction . . . 104
6.2 Internal Exploratory Design Study . . . 105
    6.2.1 Methodology . . . 105
    6.2.2 Results and Interpretation . . . 106
6.3 Applications and Guidelines . . . 109
    6.3.1 Macroalgae in the Gulf of Mexico . . . 109
    6.3.2 Brain Microstructure Imaging . . . 110
    6.3.3 Astrophysics . . . 112
    6.3.4 Abstract Data and Future Work . . . 114
6.4 Conclusion . . . 115

7 Data Streaming and Remote Rendering for 3D Scientific Visualization 116

7.1 Introduction . . . 116
7.2 Related Work . . . 119
    7.2.1 Remotely Visualizing Large Volumetric Datasets . . . 119

7.2.2 Untethered HMDs, Remote Rendering, and Latency Mitigation for AR/VR 121
7.3 Architecture 124
7.3.1 Motivation and Goals 124
7.3.2 High-Level Architecture Design 126
7.3.3 High-Level Implementation Approach 127
7.3.4 Remote Rendering Architecture 129
7.3.5 Content Frame Facade Generation 134
7.4 Performance Characterization 142
7.4.1 Data Streaming Performance Characterization 142
7.4.2 Remote Rendering Performance Characterization 143
7.5 Results 148
7.5.1 Implementation of Data Streaming 148
7.5.2 Demonstration of Data Streaming 149
7.5.3 Performance & Discussion of Data Streaming 149
7.5.4 Implementation of Remote Rendering 151
7.5.5 Demonstration of Remote Rendering 152
7.5.6 Performance & Discussion of Remote Rendering 155
7.6 Roadmap 162
7.6.1 Formal Evaluation of Performance Characterizations 162
7.6.2 Further Development 163
7.7 Conclusion 166

8 Conclusion 168
8.1 Discussion and Ongoing Work 168
8.1.1 Artifact Based Rendering 168
8.1.2 Data Streaming & Remote Rendering 172
8.2 Summary and Review of Primary Contributions 172
8.3 Vision of the Future 174
8.4 General Conclusions 174

References 177

List of Figures

3.1 Bento Box is a virtual reality visualization and 3D user interface technique for comparative analysis of data ensembles, such as this set of 10 time-varying simulations of blood flow around a cardiac lead in the right atrium of the heart. Each *column* shows a simulation with different parameters, here varying the length and stiffness of the lead. Each *row* shows a different view of the data. The top row is a zoomed-out overview. Users add additional rows of complementary, zoomed-in views during analysis. 13

3.2 The Bento Box widget arranges multiple views of volume data within a grid of "cubbies". 23

3.3 The 3D bimanual user interface is implemented as a finite state machine. There are four main states, and the system transitions between them based upon the positioning of the hands (DH = dominant hand, NDH = non-dominant hand) relative to the cubbies. As illustrated in the blue portions of the diagram, the actions triggered by the pressing the buttons on the VR wands held are different depending upon the context provided by the current state. 26

3.4 A new sub-volume of interest is created with an interactive selection. From left to right: (a) Assume the starting state is a Bento Box with two rows. (b) A click within any of the cubbies (in this case, one from the top row) using the primary button on the DH wand defines a center point for the selection (the black dot). (c) Dragging defines the size of the selection box to create. (d) After releasing the button, a new row of view settings is added to the bottom of the Bento Box. 27

3.5 A visualization design palette is used to change the variable displayed on each major graphic element of the visualization (in this case: heart walls, lead, and flow) and to adjust the color map applied to the data. A set of possible color maps to apply to each variable is loaded during initialization, and the user may change the color map to display in each row of the Bento Box interactively by dragging one of these colormaps onto a specific row. Toggle buttons at the bottom of the palette control the visibility of specific data instances (the columns), labeled with instance parameters. 28
3.6 Diagram of the simulation scenario with a cardiac lead implanted in the right atrium of the heart. 32
3.7 Bento Box arranged for a comparison of cardiac leads with three different stiffness parameters, increasing in stiffness from left to right. The top row shows an overview of the dataset. The middle two rows highlight stress on the lead itself, which appears to increase with stiffer leads. The bottom row zooms in on the attachment point of the lead in the atrial appendix, showing how in all cases the flow stagnates near the attachment point. Here, stress on the atrial walls also appears to increase with stiffer leads. 36
3.8 Bento Box arranged for a comparison of different length cardiac leads. Lead length increases from left to right. At this timestep in the simulation, the longest lead length creates a slower flow (i.e., darker red path lines). 37
3.9 Rendering frame rates decrease as additional cubbies are added to the display. Since there are multiple ways to construct a Bento Box (e.g., there are 4 possible arrangements for 10 cubbies 10×1, 5×2, 2×5, 1×10), these results are for a systematic sampling of possible configurations. The trend line is a logarithmic fit ($R^2 = 0.76$). 40
4.1 Two immersive scientific visualization projects with artist-oriented design processes. (a) Weather Report, an interactive outdoor art installation which provides an evocative comparison of objective climate data and participant-driven subjective memories. (b) Lift-Off, a existing artistic hybrid 2D/3D sketching system, applied to visualizing medical data. . 45

4.2 Visitors walk through a tunnel of animated “balloon pixels” that depict 4.5 decades of local weather data constructed from both objective measurements (right wall in this view) and the subjective memories of visitors (left wall). 48
4.3 A tunnel of balloon pixels along the walking path in Mill Ruins Park, Minneapolis, MN. 50
4.4 The data-to-visual mapping for each wall uses a hierarchical arrangement for time, decades are displayed in the two leftmost columns, with the current decade highlighted, followed by months of the year with the current month highlighted, followed by days of the month, and so on. The display animates to display the entire six decades of data over the course of the night. 51
4.5 Two frames from an animated blue rain effect temporarily superimposed over the objective weather wall. 52
4.6 Visitors enter weather memories using a three-stage, multi-touch interface on a kiosk. 53
4.7 Thousands of visitors experienced Weather Report at Northern Spark . 54
4.8 The process of designing from a sketch in Lift-Off. 57
4.9 2D medical data can be converted to line images for use with Lift-Off. Data credit: X-ray image used through Creative Commons from Majorkev on Wikipedia https://creativecommons.org/licenses/by/3.0/.) . . 60
4.10 A complete 3D model of the broken clavicle from two angles. Shown both with ((a), (c)) and without ((b), (d)) the design scaffolding. 62
4.11 Using immersive data-driven 3D annotations to explain treatment options. 63
4.12 Three variants of the robotic mechanism, sketched in the 3D immersive environment. 65
4.13 Critique of the robotic surgery device sketches can occur directly in the immersive environment. 66
4.14 Weather Report drew in visitors to not only experience a visualization of climate data, but reconsider their subjective understanding of climate as it relates to objective information. 68

5.1 Using traditional physical artistic media as input to the digital visualization pipeline provides a richer visual vocabulary and opens the door for artists to participate in creating more expressive and engaging 3D scientific visualizations. This example helps scientists understand commercially viable macroalgae growth in the Gulf of Mexico by encoding temperature and salinity from remote sensing together with eddy direction and curvature and three nitrate concentrations from computational simulation. 72
5.2 The ABR pipeline contains three main stages. 79
5.3 Artists will recognize the formal properties of (point, line, form, texture, and color) in these visual examples. Visualization scientists will recognize magnitude channels to encode ordered data and identify channels to encode categorical data. The volume category focuses on color schemes for volume rendering algorithms. 80
5.4 The EinScan-SE structured light 3D scanner makes 3D scanning of physical artifacts reliable and reproducible. 82
5.5 The Color Loom applet. Artists drag and drop source images into the left panel and pull swatches of color from these, which are then copied and arranged in the right panel to create a color map. 84
5.6 The Texture Shaper applet. A: Original source images, B: Selecting a cropping box; C: Output images and normal maps. 85
5.7 The Infinite Line applet. A: The user interface with parameter controls and texture synthesis preview. B: Examples of textures synthesized from inkwash, rice grains, and ink dots. 86
5.8 The Glyph Aligner applet. Artists use trackball controls in the left panel to align a 3D scanned glyph. The right panel provides a glyph field preview using synthetic data. 87
5.9 The EinScan-SE structred light 3D scanner makes 3D scanning of physical artifacts reliable and reproducable. 88
5.10 Example renderings and parameters for vis layers in the ABR rendering engine. 89

5.11 Options for applying vis assets to lines or surfaces include (a) color mapping, (b) data-driven texturing, (c) data-driven texturing with bump mapping, (d) data-driven texturing with blending and masking, (e) data-driven texturing with masking to create an organic line profile. 91

5.12 Two examples of surface layers using color and aspect ratio to encode multiple variables in both 3D glyph and 2D glyph styles. 94

5.13 Two examples of line layers using color and texture to encode multiple variables in both ribbon and tube line styles. 95

5.14 Two examples of surface layers using color, texture, and alpha masking to encode multiple variables. 96

5.15 An example of volume rendering using a colormap as a transfer function to encode a volumetric variable. 98

5.16 One of several results from the upcoming Chapter 6 ABR design study with the biogeochemistry data in the Gulf of Mexico [Wolfram et al. 2015]. The legend has been automatically generated from the colormaps used in the visualization. 101

6.1 Process and results from the internal exploratory design study with ABR on the biogeochemistry data in the Gulf of Mexico [Wolfram et al. 2015], left to right: pre-made glyphs; glyphs painted during the study; glyphs constructed during the study; textures captured pre-study; detail of final visualization; visualization of the Gulf of Mexico. 107

6.2 Left - Encoding options in Paraview, Right - Results from the internal exploratory design study with Paraview. 108

6.3 Visualizing brain microstructure in 3D. 111

6.4 Top row: Two variations of handcrafted multivariate volume visualization simulating water formation in the early universe. The turquoise volume rendering represents the particulate density, the water is shown in the curvilinear flowing blue forms, selected for their detail and thus becoming a focal point [Lauer and Pentar 2012]. The orange, white and green represent the three heavy metals being tracked, CH4, CO and OH respectively. The comparison shows the artists iterations needed to accurately depict the science given the complexity of variables glyphs and modifiers within the system. The right side is the earlier image. The left, the final, providing the emphasis on the water, encoded with an associative form. Bottom row: This Snapshots from the design process of assigning *elements*, *families*, and *modifiers* to the data, starting with on the left with a geometrical family typical of present-day scientific visualization, moving toward a richer, handcrafted visual language, and then using color to double-encode the type of each variable. 113

6.5 A supernova midway through its explosion. Left: A three-layer volume rendering. In related work, scientists used this visualization to for the first time see three of the variables from their simulation visualized in the same three-dimensional space. Right: A visualization using hand-sculpted families of design elements together with a single layer of volume visualization depicts twice the number of variables and uses the extended visual language to support hierarchical associations (e.g., the two styles of water glyphs are both round and smooth, the three styles of metal glyphs are all angular). 114

7.1 Three connected nodes - A supercomputer, a high-performance graphics computer, and a consumer AR/VR device - are connected by two types of transmission strategies: A) Streaming simulated data from the supercomputer onto a rendering node, and B) Streaming a pre-rendered immersive view from the rendering node to the low-cost HMD. 126

7.2 A sequence diagram showing how two data objects are transmitted from a machine running Paraview to a machine running Unity upon some change to the dataset (e.g. an incremented timestep or a new sampling parameterization). 129

7.3 a) The client viewer application contains a Unity Prefab (green & red arrows) that serves as an origin for the remoted rendered content, which can be moved dynamically by the user. The user's viewpoint relative to this Prefab is captured (blue) along with camera parameters (purple) and these are sent as a content frame request to the remote rendering server. b) A scene of Unity objects (yellow) is captured by a virtual camera (gray). This camera has camera parameters (purple) and offset (blue) relative to a Unity Prefab (green & red arrows) based on the request received from the client (a). c) The content is rendered as a buffer of pixels, plus any additional structural information required such as per-pixel depth, and returned to the client as a content frame response. d) The received content frame is used to produce a 3D "facade" of the original content, displaying only the features of the content visible from the requested viewpoint. This facade is rendered interactively relative to the client Prefab until a new content response is received. 130

7.4 (a) The settings required of a developer to add a Remote Render Server prefab object to any Unity Scene. The developer must only specify a port on which the server will listen for new Client connections. (b) The settings required of a developer to add a Remote Render Client prefab object to an AR/VR viewer application. The developer must specify a port, IP Address, and a list of dynamic viewpoints from which to render. In this figure, the left and right eyes of a VR Player Rig have been selected and labeled respectively. 132

7.5 Specifying layers to be captured for each view in the client viewer application 134

7.6 A Unity scene of objects on the Remote Rendering Server application . 135

7.7 A grid of quads aligned at the far-plane specified by the configuration in the content frame response. Here the resolution of the grid has been specified as 32x32 for the purpose of illustration, but in practice grid may be more densely subdivided. 136
7.8 a) The mesh is deformed by displacing each vertex towards the content frame's viewpoint according to the sampled depth. b) The RGB components of the texture are projected onto the mesh from the content frame viewpoint. c) The rendered result from the requested viewpoint. d) The rendered result from a different viewpoint offset towards the left. . . . 137
7.9 a) Before displacing by depth, the quads of the mesh are duplicated when overlapping a depth discontinuity over a specified threshold. b) The RGB components of the texture are projected onto the mesh, sampling the depth component to determine whether the sample pixel belongs at the corresponding mesh position, or discarded. c) The rendered result from the requested viewpoint. d) The rendered result from a viewpoint slightly off-set to the left. 139
7.10 a) The results of smear reduction with quads on the far-plane removed. b) The RGB image projected only on the remaining mesh. c) The rendered result from the requested viewpoint. d) The rendered result from a viewpoint slightly off-set to the left. 140
7.11 a) Two meshes placed at the far-planes of two offset viewpoints. b) The two RGB images projected onto their respective meshes and rendered coincidentally. c) The rendered result from the right-most viewpoint. d) The rendered result from near the left-most viewpoint. 141
7.12 Two images taken by photographers in Austin during an NSF event unveiling a new supercomputer called Frontera. At this event we presented a demo of live data streaming from Frontera to a laptop running an ABR visualization. 150
7.13 An ABR visualization being remotely rendered into a Magic Leap to be viewed with a 3D printed bathymetry. 152
7.14 Our remote design discussion about an asteroid impact dataset being viewed on many types of devices concurrently. 153

7.15 The visual results of adjusting two key remote rendering parameters, the pixel resolution of the RGBD content and the vertex-resolution of the facade mesh. For each parameter, three values were selected. Note that when the facade resolution is low, blocky gaps are more likely to appear, and glyphs have a higher chance of disappearing. 157

7.16 Varying degrees of distortion from an early stage of the remote rendering project. (a) Small amounts of smearing between glyphs and the background when the user has moved a few centimeters from the requested viewpoint. (b) Large amounts of smearing when the user has moved about 90 degrees around the scene from the requested viewpoint (an extreme example). 159

7.17 (a) The result of adding a locally-rendered room environment with smear removal integrated, leaving silhouette holes in occluded geometry rather than 3D smears. (b) Additional artifact reduction by using separate remote rendering views for Surface and Glyph layers in Unity 160

8.1 A prototype for what an automatically-generated legend could look like, with hierarchical drill-downs for seeing more detailed multi-variable encodings. 171

Chapter 1

Introduction

Today, due to advances in technology, the data we wish to visualize are more complex than ever. Even with the most cutting-edge computers and software, scientists are quickly reaching the limits of what their visualization tools can support as they seek to answer important questions, and present their progress to the public. Some scientists are developing computational tools to leverage high-field Magnetic Resonance Imaging (MRI) for understanding structural and functional alterations of brain connections in neuro-degenerative disorders. Other scientists are working with massive computational simulation models of ocean currents and biochemistry to study mariculture, specifically investigating which regions of the earth's oceans support commercially viable microalgae growth as a sustainable source of biofuel. Complex data sets such as these can be as challenging to understand as they are to collect.

As scientific data sets get more complex, the task of transmitting the relevant information into the human mind becomes more difficult. This challenge manifests itself both when scientists are pursuing new discoveries in exploratory visualizations of highly complex data sets, and when visualizations are curated for a public audience via expository visualizations. While studying the complex connections in the brain, scientists wish to discern the relationships between the directions of several crossing pathways in the white matter areas as they consider multiple microstructural parameters, such as fiber density and axonal diameter. And as research in areas like these continues, scientists must find ways to expose their complex progress to the public through engaging and informative presentations — establishing a compelling connection with their audience — as they seek to educate and gain support for continued exploration. Producing visualizations to effectively enable understanding in any of these cases requires careful representation of several types of information at once.

Designing visualizations that successfully display numerous data fields simultaneously in a discernible way is a research problem that the field has struggled to to answer over the past 20 years; in fact, this problem has been recognized by several as a “top 10” research problem in visualization [Johnson 2004]. Visualization is, itself, the process of selecting meaningful visual metaphors for encoding data to make information easily perceived. The most effective visualizations utilize an advanced understanding of human perception, presenting information and relationships in an image that can be accessed by a viewer. When designing a visualization for the effective analysis of neural pathways in the brain, careful choices need to be made for each of the important variables: Should color be used to reinforce the directional information? Can the axonal width variable be accurately communicated by simply making the visualization elements wider? How many sample points can be displayed with 3D glyphs before the proverbial forest is obscured by the trees? Is it appropriate to combine both directional glyphs and interpolated fiber lines in the same volumetric space? To design effective visualizations is to leverage an understanding of intuitive meaning and narrative, and to wield a discerning eye for color, texture, shape, and form. While these steps are essential for producing an effective data visualization, the related skills are not necessarily practiced by or taught to the scientists or computing practitioners who most often find themselves designing visualizations.

1.1 Art and Science

In many ways, the visualization design task parallels the highly-skilled work of visual artists. For centuries, artists have been using their hands to produce artifacts that help us see the world in new ways and make sense of abstract ideas. Through use of colors, textures, shapes, and forms, artists craft 2D and 3D works that express an often complex concept or story in such a way as to be considered and discussed by people of all backgrounds. During the Renaissance, art and science were intertwined as scientific studies provided new inspiration for art, and artistic illustration provided incredibly detailed means of documenting new scientific discoveries. And even today, artists are inspired by science as they work to portray the same marvels, mysteries, and monitions as those which scientists investigate. It stands to reason, then, that the dictionary definition of visualization, "the act or process of interpreting in visual terms or of putting into visible form," describes works of art just as much as it describes scientific data visualizations.

This connection between art and science has not gone unnoticed. The visualization research community has often looked to artistic techniques such as color theory, illustrative style, and oil painting layers. These techniques have been studied by computer scientists and distilled into non-photo-realistic rendering algorithms for automatic data encodings [Healey 2001; Healey and Enns 2002; Kalnins et al. 2002; Kirby et al. 1999; Laidlaw et al. 1998; Tateosian et al. 2007; Winkenbach and Salesin 1994]. The result is often evocative and engaging beyond the otherwise sterile computer graphic aesthetic of traditional interactive scientific data visualizations. However, these "stylized" visualizations are usually produced synthetically, without the involvement or intervention of artists themselves.

Even more exciting things can happen when artists and scientists come together and align their interests and expertise through collaboration on scientific visualizations. Donna Cox refers to such a partnerships as a "Renaissance Team," in which a team of interdisciplinary specialists come together to solve research problems [1988]. We see several clear advantages to facilitating Renaissance Teams of scientists and artists to create scientific visualizations:

- Scientists provide new insight into challenging and important issues which can serve as sources of inspiration for artists.
- Artists are skilled at lateral thinking and can aid in re-framing ideas to shed new light onto scientific problems.
- Scientists rely on visualizations to detect and analyze both large-scale patterns and small details, and artists are trained to select colors, textures, shapes, and forms for drawing the eye to both.

- Visual art naturally captivates all types of viewers — whether the viewer is searching for meaning or dwelling on a particular emotion or idea — and science could benefit from a similar level of engagement.

However, inviting artists directly into the process of designing interactive scientific visualizations is difficult. While a few specialized techniques have been demonstrated for placing visualization design into an artist's hand, there are as of yet no fully integrated workflows for artists to take on the design role in volumetric[1] visualization at-scale. We see several specific obstacles in the way of achieving the aforementioned advantages in the design of volumetric scientific data visualizations:

- There are strong language barriers when it comes to communicating between scientists and artists, both in terms of the scientists' data to be visualized, and in terms of the artists' techniques.
- Existing 3D scientific visualization software does not present user interfaces conducive to an artist's iterative creative process, and is often built to only be accessed in an advanced research lab.
- Current techniques tend to constrain data encodings to a narrow set of visual elements that have been tested and vetted by the visualization community, further restricting freedom to try bold new visual metaphors.
- Most visualization encodings are produced through entirely synthetic computer graphics algorithms, vastly reducing the visual vocabulary of color, texture, shape, and form with which traditional artists are familiar.

1.2 Palpable Visualizations

We overcome these obstacles and provide a new framework for facilitating collaborations between scientists and artists in order to produce more discernible and engaging data visualizations. We call these Palpable Visualizations. The word palpable has two important definitions: (1) Easily perceptible by the mind, and (2) Capable of being touched and felt. Definition 1 is both a desirable characteristic for a scientific data

---

[1] "Volumetric," referring to the inherent spatial, 3D nature of the visualized data.

visualization meant to be analyzed as well as a quality of well-designed visual art. Definition 2 is a characteristic of art that naturally engages a museum audience to consider and relate with a piece of artwork (whether touching is actually permitted or not), and this accessibility will serve as a basis for providing deeper engagement with Palpable Visualizations. Thus, we define the two Pillars of Palpable Visualizations:

1. **Discernible:** Palpable Visualizations should maximize the amount of information being displayed and minimize the effort required to understand the information being displayed.
2. **Accessible:** Palpable Visualizations should be both designed and presented in such a way as to encourage the participant to relate with the information being displayed at a physical level as one would relate with a museum installation. These visualizations should also be targeted for widely-available immersive display.

To produce Palpable Visualizations, one of our key proposals is to create new tools and techniques for grounding the entire visualization design process in the physical world. By bringing the design process out of the algorithmic domain and into the physical, the door is opened for traditional artists to lend their hands to the visualization design task. New paradigms for visually encoding scientist-selected data with artist-selected colors, textures, shapes, and forms help bridge the language barrier between the two areas of expertise. New physical-digital hybrid workflows enable for artists to rapidly prototype encodings as artifacts in the physical world and test them out in a digital design environment. New artifact-based rendering techniques enable the final encodings to closely resemble the original artist-crafted physical artifacts, harnessing preexisting associations with physical forms. And new VR interaction and rendering techniques enable fluid and natural exploration of complex volumetric scientific data, even on affordable devices that can be accessed outside of advanced research labs by both scientists and the public.

1.3 Thesis Statement

This brings us to the thesis of this dissertation:

Interactive computational workflows that equip artists to fluidly design scientific visualizations and enable any participant to immersively experience and relate with the results can lead to visualizations that are simultaneously more discernible and more accessible.

1.4 Contributions

The key contributions of this dissertation are:

- An assessment of the current state of the art of immersive scientific visualization through a challenging case study of ensemble visualization of 4D simulations of blood flow through the heart.
- Two complementary case studies of studies that apply art and design methods to exploring and presenting scientific and medical data.
- A theory and implementation of Artifact-Based Rendering, a framework of techniques for enabling artists to create Palpable Visualizations
- Results of applying Artifact-Based Rendering to 3 specific 3D datasets including biogeochemistry, brain microstructure, and astrophysics
- An implementation and characterization of remote rendering from a high-performance rendering computer to a consumer untethered VR display with parallax correction techniques and collaborative multi-user support

1.5 Overview of this Dissertation

In this dissertation, relevant **related work** is presented in Chapter 2. More in-depth related work may be found in chapters 3, 4, 5, and 7.

State-of-the-art Visualization Design

In Chapter 3, our new immersive user interface called **Bento Box** allows fluid comparison of complex 4D engineering simulation ensembles in a research VR CAVE environment. This work serves as an example of building cutting-edge immersive scientific visualizations for discernibility with the state-of-the-art in current visualization techniques, and also as a study in the limits in accessibility we hit while designing traditional computer graphics applications.

Step Back to Examine More Accessible Design

Having learned some new things through our study of Bento Box, Chapter 4 takes a step back to evaluate more fluid creative processes and strategies for broadening participation (both in design and display) through two works, **Weather Report** and **Lift-Off**.

For Weather Report, we collaborate with architects in their own design methodologies to produce an interactive outdoor art installation which provides an evocative comparison of objective climate data and participant-driven subjective memories. And with Lift-Off, an existing artistic hybrid 2D/3D sketching system is adapted for collaborative design and discussion in two medical case-studies in which engineers and physicians are free to explore and communicate with 2D and 3D sketching.

A Framework for Designing Palpable Visualizations

Chapter 5 details a framework enabling Artists to create Palpable Visualizations inspired by the observations made in the previous two chapters. This chapter presents recently published work on a new framework of **Artifact-Based Rendering (ABR)** techniques with which artists can produce immersive 3D visualizations crafted entirely of aesthetics collected from the physical world. And Chapter 6 demonstrates the power of ABR through a design study and several examples of applying ABR to diverse 3D datasets.

Increasing Accessibility via Remote Rendering

Chapter 7 presents a final parallel step towards increasing the accessibility of palpable visualizations by providing a **Data Streaming and Remote Rendering** system for transmitting complex immersive visualizations to affordable head-mounted-displays from remote high-performance rendering computers and super computers.

Chapter 2

Related Work

2.1 Making Visualizations Discernible

Developing new and effective data visualizations is always challenging. And when it comes to immersive visualizations of complex volumetric datasets, a critical goal is maximizing the discernibility of the encoded data.

A specific example explored in Chapter 3 is designing a visualization to support comparative analysis between several related data instances is an area of diverse research approaches. Sedlmair et al. present a conceptual framework that is useful for characterizing ensemble visualizations, including several fundamental approaches for navigating through a parameter space [2014]. One such approach is “local-to-global”, as demonstrated in Design by Dragging [Coffey et al. 2013]. Here, for the most part, the user focuses on visualizing just a single instance of data at a time, and the emphasis within the user interface is on making it easy to transition from the current instance to others that make up the ensemble. Connecting to the literature on comparative visualization, this work uses what Kim et al. describe as an “interchangeable approach” augmented with animated transitions [2017]. This approach has a key theoretical perceptual limitation—the data to compare are not simultaneously visible.

A finer level challenge in visualization design is supporting the discernibility of individual variables within a data instance. This includes problems such as designing effective colormaps for encoding scalar data ranges, or using texture to encode information across surfaces. Seminal research on designing and testing perceptually accurate colormaps for general use in visualizations [Moreland 2009; Rheingans 2000; Rogowitz and Kalvin 2001; Ware 2012; Zhou and Hansen 2016] establishes several rules of thumb, such as relying primarily on luminance and saturation for depicting magnitude data. Tools are also available for evaluating and modifying maps to adhere to commonly accepted guidelines [Bujack et al. 2018; Kovesi 2014; Zhou and Hansen 2016], which closely parallel the fundamentals of color theory as studied by artists [Albers 2009; Itten and Birren 1970].

Varying texture in response to data is a technique that has also been used previously [Healey and Enns 1999; Laidlaw et al. 1998; Ware and Knight 1995], including to encode uncertainty [Botchen et al. 2005]. Closely related to our work is that of Interrante et al. who encoded data using natural 2D textures of fibers and weavings of different densities [2000]. Later Gorla et al. extended this to synthesize texture from an example that follows a vector field on a 3D surface [Gorla et al. 2003]. We identify data-driven synthesis as important step for future work.

2.2 Creative Design

This dissertation calls for the exploration of next-generation creative visualization design tools, with a focus on AR and VR for complex volumetric datasets. Such tools will allow artists to create new visualizations in a fluid, creative manner, and to design and aggregate libraries of visual metaphors to support future visualization efforts. The field of visualization is already rich with research into creative interfaces and artist-inspired techniques.

Design is an inherent challenge when crafting any data visualization, as encoding data into a visual form that effectively communicates meaning requires careful consideration [Cox 2006]. Thus, the visualization community has often turned to the field of creative design for inspiration as metaphoric mappings are needed. Healey's techniques seek to imitate painterly styles to create data visualization images [2001], Kirby focuses on the flowing strokes of oil painting to bring life to visualizations of fluid mechanics [1999], Interrante and Grosch took advantage of perceptual insights used by artists when applying Line Integral Convolution to visualize the direction motion of complicated 2D flow data [1997], Joshi and Rheingans propose using the work of cartoonists to inspire new methods of time-varying data visualizations [2005], And Ma outlines several researchers' approaches to using artistic non-photorealistic rendering methods to communicate shapes and spatial relationships [2002]. These techniques demonstrate that art is an excellent inspiration for visualization design.

While these ideas are a few of the ways that technical designers and scientists are indirectly learning from artists, other research endeavors to bring artists themselves into the visualization design process. One key work in this area is that of Cox, who instigated "renaissance teams" of artists and scientists through special software that enables this collaboration, arguing that artists possess inherent talents that are increasingly important as technology requires more and more "seeing" of complicated data [2008]. Many more tools have been invented to this end. Mitchell, Ware, and Kelley proposed and evaluated a method of human-in-the-loop parameter optimization allowing artists to craft a visualization through interactive feedback, and showed that trained artists were able to produce better results than non-artists [2009]. Bruckner and Gröller took on the difficult issue of specifying volume rendering transfer functions with VolumeShop, an interactive 3D illustration environment for directly manipulating the appearance of volume renderings [2005]. Later, Bruckner and Gröller developed another interface for the specification of transfer functions, allowing artists to define the appearance of the profile of volume rendered data using halos, similar to common lit-sphere techniques [2007]. What-You-See-Is-What-You-Get (WYSIWYG) is another form of direct-manipulation, and Guo et al. have developed a volume data visualization tool that allows artists to input sketch strokes to create transfer functions for volume rendering that offers a great degree of control [2011]. More recently, Schroeder and Keefe have presented Visualization by Sketching, an interface designed explicitly for artists to paint directly onto 2D data visualizations to create color maps and glyphs; they were able to successfully reproduce several classic data visualizations using only the artistic interface [2016]. These techniques are paving the way for artists to become directly involved in the visualization design process, and for inspiring scientists to think more creatively when communicating their research.

However, each of these creative design tools are developed for a specific type of data. Bruckner and Gröller's work, along with that of Guo et al., are tightly focused on volume data rendering. Furthermore, they only provide interaction on a traditional desktop computer workspace. Schroeder and Keefe took another step forward by providing an interface based on the tools with which an artist is already familiar – namely, a stylus-based sketching interface – for designing visualizations of both 2D scalar and vector fields in the same space. While this work propels the research of creative visualization design forward, it falls short of providing fluid interfaces for working with the 3D nature of complex volumetric datasets.

Samsel et al. have developed ColorMoves [2016], a design-oriented tool focused on creating colormaps tailored to any type of data for a wide range of analysis tasks. Accessible to both artists and scientists, ColorMoves provides a fluid 2D interface for tailoring new colormaps to the distribution of a data variable using a wide range of artist-curated colormaps. While ColorMoves is only supplemental (it requires exporting a still frame from a data visualization system into a web interface), it is an excellent example of one important piece of a creative visualization design tool: producing new visual elements.

This dissertation looks at visualization techniques that allow the user to fluidly manipulate and customize a 3D visualization, agnostic to the types of data being displayed. Furthermore, this dissertation proposes a unified experience for visualization design across many types of data, collecting many of the currently isolated techniques listed above.

2.3 Grounding Visualizations in the Physical World

Artists bring to the table a heightened emphasis on the connection data makes with the people who view it. By opening visualization design to artists and their extensive understanding of how humans experience the world around them, we hope that data visualizations can be empowered to play a more intimate and impactful role in people's lives.

One direct way that data visualization is becoming more engagingly connected with the world around us is through the practice of 'data physicalization" [Jansen et al. 2015]. Data physicalization is the process of visualizing data via a physical output (3D printouts, sculptures, active touch tables, etc.).

Often, physicalization for scientific data visualization takes the approach of starting with complex computer-based spatial datasets and 3D printing them to make them more physically accessible to audiences. One such example is Djavaherpour et al.'s work on 3D printing geospatial data to help viewers better understand the structure of a number of geographically-located types of data [2017].

But an inverse approach can also be taken. Rather than starting with computer models and manifesting them directly in the physical world, another kind of accessibility can be achieved by starting with recognizable real-world artifacts and building up data encodings with them. For example, artists like Mielbach [Samsel 2013] use physical materials to provide context and connection for science through hand-crafted physical data visualizations. Both visual artists and psychologists have studied the geometric characteristics of shape (e.g., roundness, angularity, simplicity, complexity) and their impacts on human emotional responses [Lu et al. 2012]. There is evidence that data visualizations can be more effective and engaging when created by hand. Route maps can be more effective when presented in a hand-drawn style [Agrawala and Stolte 2001], and hand-draw iconography improves engagement and retention in data-driven storytelling [Lee et al. 2013; Rogers et al. 2017].

As we will discuss, grounding work in the physical world is one of our key strategies to make visualization design accessible to designers and to make the visualizations themselves accessible to a wide audience.

Chapter 3

Bento Box

Understanding the Current State-of-the-Art for Immersive SciVis

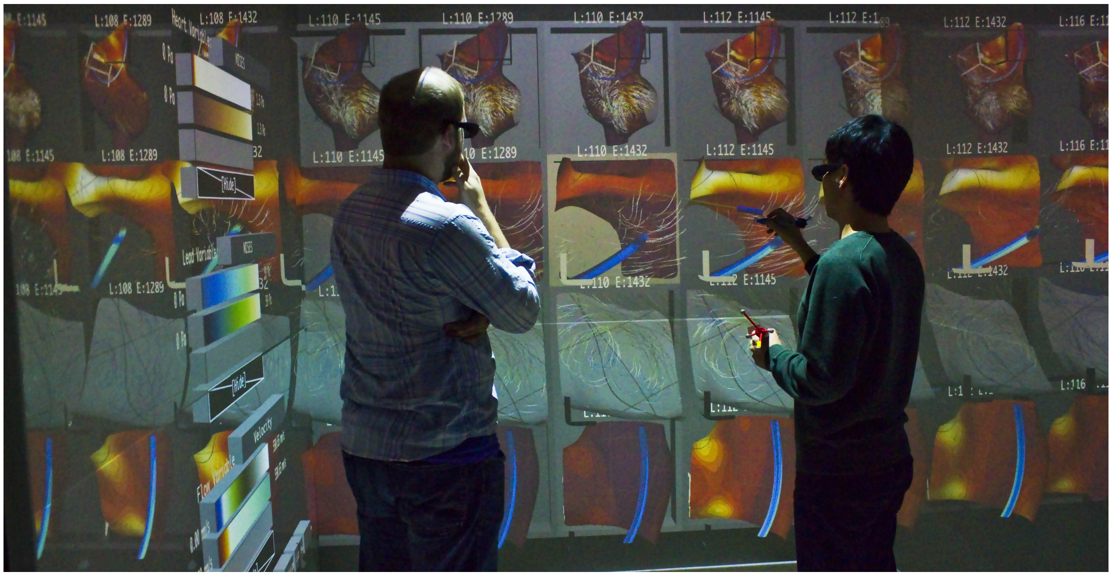


Figure 3.1: Bento Box is a virtual reality visualization and 3D user interface technique for comparative analysis of data ensembles, such as this set of 10 time-varying simulations of blood flow around a cardiac lead in the right atrium of the heart. Each *column* shows a simulation with different parameters, here varying the length and stiffness of the lead. Each *row* shows a different view of the data. The top row is a zoomed-out overview. Users add additional rows of complementary, zoomed-in views during analysis.

3.1 Introduction

[1] Scientific data visualization is an active research area with a large body of literature regarding best practices and techniques. To establish a base-line for modern scientific visualization, we look at a project that simultaneously relies on tried-and-true visual design and novel interaction techniques. This chapter presents work of a large immersive visualization system called Bento Box [Johnson et al. 2019a] as an example of the sort of complex 3D data that palpable visualizations need to be able to support. [2] In the conclusion at the end of the chapter, we will consider the limitations of a traditional graphics application like Bento Box as it pertains to discernibility and accessibility.

Science and engineering workflows increasingly rely upon ensembles—"concrete distributions of data, in which each outcome can be uniquely associated with a specific run or set of simulation parameters" [Obermaier and Joy 2014]. Analyzing these ensembles is a challenging task that involves not just understanding specific data values and trends but also making comparisons. Visualization can help, and recent ensemble visualization research has made it possible to: (1) manage and render some of the large datasets that are encountered with ensembles [Vohl et al. 2016]; (2) use interactive techniques to navigate through large ensemble parameter spaces [Sedlmair et al. 2014], including using both local-to-global [Coffey et al. 2013] and global-to-local approaches [Bruckner and Moller 2010]; and (3) use simulation steering to explore "what if" scenarios [Waser et al. 2010, 2014]. Unfortunately for scientists and engineers, much work remains—successful ensemble visualization requires not just a minor adjustment of the traditional visualization pipeline but rather a significant reworking. Major current problems include:

- The lack of connection between the research on ensemble visualization and theoretical research on comparative visualization [Gleicher 2017; Gleicher et al. 2011; Kim et al. 2017] which discusses fundamental trade-offs between perceptual strategies

---

[1] This chapter is based on work published in Frontiers [Johnson et al. 2019a].

[2] Acknowledgements: Bogdan Tanasolu, Georgi Subashki, and Shan Sandy Wang assisted with developing simulations and data wrangling. Dr. Paul Iaizzo and the University of Minnesota Visible Heart Lab provided access to heart geometry data and feedback during iterative development. This work was supported in part with computing resources from the Minnesota Supercomputing Institute and hardware donations from NVidia. The software utilizes the VRPN library maintained by UNCChapel Hill's CISMM project with support from NIH/NCRR and NIH/NIBIB award 2P41EB002025

required for making comparisons, such as *juxtaposition (side-by-side), superposition (overlayed), interchangeable (animating through or interactively switching between viewing a single data instance at a time), explicit encoding (e.g., computing the difference between data instances), and hybrid approaches.*

- The lack of ensemble visualization techniques, including user interfaces, designed specifically for use in virtual reality (VR) environments. We know perspective-tracked, stereoscopic displays can outperform desktop tools for spatial perception tasks [Ware and Mitchell 2005, 2008], but designing effective VR visualization tools is challenging and requires synthesizing and refining user interface research results on bimanual interaction [Hinckley et al. 1997], navigation [Stoakley et al. 1995], selection [Bowman and Hodges 1997], and manipulation [Mapes and Moshell 1995] (citations limited here to some early, seminal works); these VR and 3D user interface research results are not always widely cited and used in scientific visualization.

- The need for additional examples (i.e., case studies) of how to organize data and perform rendering of different types of ensembles. This is important because rendering at the ensemble scale requires fundamentally different approaches for 4D fluid dynamics computed on unstructured grids (explored here) as compared to, for example, 2D maps and imagery [Javed et al. 2012]. In addition, visualizing spatial relationships and interaction between data parts in multimodel scenarios (i.e. fluid and structure interactions) is rarely explored [Kehrer and Hauser 2013].

This chapter addresses the specific unsolved challenge of visualizing moderate-sized ensembles (e.g., containing on the order of 10 data instances) of state-of-the-art, time-varying fluid-structure interaction simulations run on high-performance computing platforms. This size of ensemble is useful to study because it is large enough to present challenges in rendering and visual comparison but not so large as to rule out the possibility of visualizing the entire ensemble simultaneously. The strategies developed for this scale can likely be combined with others (e.g., filtering) to address larger ensembles.

This chapter also focuses on VR-based visualization. The rationale for VR is based on the data. With interdisciplinary collaborators, we are studying simulations of blood flow through the heart, and this requires analyzing complex spatial relationships, such as subtle differences in 4D vortical structures. Formally, low-level perceptual studies suggest that perspective-tracked, stereoscopic visualization can facilitate understanding complex spatial relationships found in 3D data [Ware and Mitchell 2005, 2008], providing evidence for the likely utility of VR in our work. Informally, our collaborators have consistently cited an improved ability to see spatial patterns in the data with VR and repeatedly demanded to analyze the data using VR over the course of a 5+ year project. Taken together with the fact that there is no longer a financial barrier to using VR for scientific visualization, we take this as strong motivation.

Our proposed solution builds upon recent theory on comparative visualization in 2D contexts [Gleicher 2017; Gleicher et al. 2011] as well as 3D and 4D (3D + time) contexts [Kim et al. 2017]. Specifically, we adopt the fundamental approach to visual comparison known as *juxtaposition* and adapt it to suit VR-based volumetric visualization.

The rationale for the juxtaposition strategy as compared to the *interchangeable* strategy (another fundamental approach discussed in the literature) can be summarized by the visualization rule of thumb, “eyes beat memory” [Munzner 2014]; making comparisons is easier if we can see the items to compare simultaneously rather than trying to remember one or more previously viewed items. The rationale for juxtaposition as compared to *superposition* is specific to the data of interest. These 4D data are so dynamic and the spatial patterns so complex, that we rule out superposition due to the extreme complexity and occlusion issues that would occur when rendering 10 blood flow datasets in the same visual space. The final fundamental approach to comparative visualization discussed in the literature is *explicit encoding*. Explicit encoding could be used within an extended version of our tool (future work); however, designing a new explicit encoding for comparison is so dataset specific that it often becomes its own research project with results that may not translate well to other datasets.

This leads us toward a juxtaposition approach. Juxtaposition, however, presents its own trade-offs, which make designing an effective ensemble visualization challenging. For example, one concern with juxtaposition is that when visuals to compare are viewed side-by-side, the viewer’s eyes must move back and forth between the visuals in order to find correspondences and notice differences. This takes effort and time, and the naïve approach of simply rendering each data instance, one next to the other, is unlikely to be the most useful, especially when the key differences are subtle and appear in small sub-regions of the volume data. Our strategy to mitigate this is to make it possible for users to interactively design a spatial layout that places all of the volumetric features of interest as close as possible to each other. We call the resulting tool, which neatly slices and places data into an organized grid of sub-volumes, *Bento Box*.

Figure 3.1 shows Bento Box in use with the medical device design application. We know engineers need to analyze several important sub-volumes of data within the right atrium of the heart, including: (1) the stress in the right atrial appendix, (2) the speed of the flow in the main vortex that forms, and (3) the stress through a cross-section of the lead inserted in the heart. Engineers must analyze all of these aspects and more, making comparisons across each instance, in order to completely understand the ensemble. A key design goal is, therefore, to make it as easy as possible for users to compare a specific volumetric feature (we will call this a sub-volume of interest) while also switching focus easily back and forth between several sub-volumes of interest. The Bento Box technique accomplishes just this.

Bento Box naturally draws upon and advances research that is relevant to both the VR and Visualization research communities, and we expect this work to be especially relevant to the growing number of researchers that span the two communities. Our contributions necessarily span both technical areas and include:

- The design of Bento Box, a VR visualization technique for comparative analysis of 4D data ensembles.

- A novel bimanual 3D user interface that supports: (1) zooming and reframing the Bento Box, (2) selecting sub-volumes of interest, (3) navigating within these sub-volumes, and (4) specifying multiple critical times to compare.

- A case study that describes how to apply rendering and data-sampling algorithms for visualizing Fluid-Structure-Interaction (FSI) simulation data for the first time to multiple instances of these data simultaneously in stereoscopic, head-tracked VR.

- A performance evaluation of the rendering and data sampling strategies applied to the cardiac application.

- User feedback from the interdisciplinary application.

The chapter is organized as follows. After additional discussion of related work, we present the Bento Box technique. The key novel aspects here are the visual layout and VR user interface, both of which are likely to generalize to use with other volumetric datasets. Then, we present the application developed as part of a multi-year interdisciplinary team-science research project; this includes important details, such as data sampling strategies, that are required to make the technique practical for use with modern, actively researched 4D flow datasets. The real-world application serves as an initial evaluation of the technique, and both user feedback and rendering performance are reported. Finally, we discuss limitations and future work as well as conclusions.

3.2 Related Work

Several areas of related work are relevant.

3.2.1 Ensemble Visualization & Comparative Visualization

Sedlmair et al. present a conceptual framework that is useful for characterizing ensemble visualizations, including several fundamental approaches for navigating through a parameter space [Sedlmair et al. 2014]. One such approach is “local-to-global”, as demonstrated in Design by Dragging [Coffey et al. 2013]. Here, for the most part, the user focuses on visualizing just a single instance of data at a time, and the emphasis within the user interface is on making it easy to transition from the current instance to others that make up the ensemble. Connecting to the literature on comparative visualization, this work uses what Kim et al. describe as an “interchangeable approach” augmented with animated transitions [2017]. Recall, this approach has a key theoretical perceptual limitation—the data to compare are not simultaneously visible.

Bruckner and Möller present an alternative [2010], characterized by Sedlmair et al. as “global-to-local”. Here, the visualization system starts with an overview in the form of thumbnail images. The thumbnails enable juxtaposed comparative visualization at the overview level. Interactive filters then make it possible for users to narrow the search space to the most promising subset of the ensemble for closer inspection. Individual data instances can then be examined one at a time in a single 3D view window. In comparing with Bento Box, a major strength of this approach is the filtering, which enables the technique to scale to larger ensembles. However, a drawback relative to Bento Box is that detailed analyses of the final volume data are done on the desktop with a single 3D window at a time. Comparing this as well as related ensemble visualization tools that include some form of juxtaposed comparison, for example World Lines and follow-on systems [Waser et al. 2010, 2014], a key difference with Bento Box is supporting not just comparison via overview thumbnails in the top row of the widget, but also detailed comparisons of even subtle variations across multiple volumes that can all be viewed simultaneously from multiple vantage points in VR.

Vohl et al. [2016] have done some of the most impressive work in ensemble visualization from a systems perspective, multiple data instances may be displayed juxtaposed for collaborative analysis using an ecosystem of displays (desktop, mobile, and VR); in VR/large-screen mode the system uses a handheld tablet as a display controller. In contrast, Bento Box is optimized for VR and introduces a bimanual user interface for not just rotating data and assigning data instances to specific sub-displays but also for selecting sub-volumes of interest in 3D space and reframing the virtual display for comfortable viewing. This is intentionally designed to enable a fluid style of data exploration so users do need to look away from the data when operating the interface.

Chi et al. [1997] introduced a spreadsheet-inspired layout for data visualization, highlighting "Custom Tabular Layouts Enable Comparisons" and including data processing operators, such as subtracting one data instance from another. Working with volumetric data where explicit "difference" encodings are often more complex to compute, Jankun-Kelly and Ma [2001] combine volume rendering and a spreadsheet-like interface to visually explore an ensemble. Axes may represent ensemble parameters, time, color maps, or transfer functions. Bento Box extends the concept of a 2D grid, small-multiples-style layout to one where users to interactively set the views along the vertical axis based on spatial croppings of volume data. In addition, Bento Box reinterprets the approach as a native VR visualization technique with integrated 3D user interface.

Finally, Alabi et al. [2012] take an innovative approach to comparative visualization where multiple surface models are sliced and then displayed in an interleaved 3D space. The result is visually distinct, but the approach shares a similar underlying theoretical reasoning with Bento Box. Both techniques prefer juxtaposition to superposition, reasoning that the resulting 3D visual display would be too complex with superposition. However, both also recognize that the spatial separation that comes with a naïve application of juxtaposition (one complete dataset next to another, then another, ...) is also problematic. The solution for both involves slicing the data, placing corresponding sub-volumes as close together as possible. The key differences in Bento Box are the use of more complex 4D data, including fluid flow, which would likely not work well with the extreme spatial interleaving Alabi et al. use for surfaces, and the fact that the sliced display can be defined interactively by the user directly within a VR environment. Slicing to pull out subsets of data for comparison has also been used in 2D visualization (e.g., for time-series and image data) [Javed and Elmqvist 2010; Javed et al. 2012]; but, the spatial arrangement and interface are necessarily different when working with volume datasets.

3.2.2 Flow Visualization & Animation

There is a long history of visualizing fluid flows in VR, dating to Bryson's seminal Virtual Wind Tunnel [1996]. Early work in this area relied upon interactive visualization widgets, such as interactively placed particle emitters and streamline rake widgets. This elegantly enables user exploration but it also has a key limitation in that the data are essentially hidden from the display until the user places widgets to reveal them. The Particle Flurries technique takes an opposite approach, using a fleet of carefully seeded particles to present a synoptic animated view of the data [Sobel et al. 2004]. For many tasks (e.g., gaining an overall understanding of a flow), we see this as the preferable approach; however, it is not well suited to comparative visualization, as reading 10 animated flow visualizations side-by-side would simply be impossible from a perceptual standpoint. Our approach is a hybrid. Flow visualizations are pre-populated with 3D comets inspired by Mitchell et al.'s carefully designed 2D streaklets [2009], but the default presentation is static rather than animated.

The best role for animation is something that was considered carefully in Bento Box. Prior perceptual research suggests that, although animation can be useful for explaining trends, it is less useful for data analysis tasks [Robertson et al. 2008; Tversky et al. 2002]. For identifying trends, it is often the case that a carefully designed static visualization is better than an animated visualization. In addition, prior VR-based studies suggest that interactive control over time is useful and important to support some tasks, such as identifying the exact moment of collision by two objects in 3D space [Coffey et al. 2012a]. These motivate our approach to default to static views of multiple time steps and support interactive control of time and animated overviews as options. A timeline interface is used; however, the direct manipulation interface introduced by Hentschel et al. [2008], would integrate perfectly into Bento Box and is a planned future addition to the tool.

3.2.3 Bimanual & 3D User Interfaces

Navigation, object selection, and object manipulation are major topics within the VR and 3D user interface research [Bowman et al. 2004]. What is unclear from the prior literature is how best to move these techniques from user studies or other application scenarios to complete, high-end visualization applications. For example, there is no prior work that demonstrates how to select sub-volumes of interest within 3D visualizations using the same bimanual controllers that are also used for multi-scale navigation of multiple datasets and for system control (e.g., interacting with menus and timelines).

Bento Box extends the bimanual scene navigation and object manipulation techniques first introduced by Mapes and Moshell [1995], which have since been revisited and revised many times (e.g., [Cutler et al. 1997; Keefe et al. 2001]). Specifically, Bento Box makes it possible to use these techniques at multiple scales. When the user's hand enters the Bento Box, the technique transitions from a scene-level manipulation to a local object manipulation interface. Another extension is that a laser (ray-casting) based selection is used for easy, coarse selections when working from a distance, but this seamlessly transitions to a cursor, point-based interface for detailed selection of sub-volumes of interest when the hand is held inside the Bento Box. In addition to these smooth interface transitions, Bento Box also includes automated view adjustments for quickly zooming to a subsection of the grid or zooming back out to an overview state.

Building upon early theoretical work in bimanual user interfaces in 2D [Hinckley et al. 1997; Leganchuk et al. 1998] operations are assigned to the hands following the guidance that the non-dominant hand should set the context within which the dominant hand performs, often more precise, operations. Prior work in VR has used these interface design concepts [Keefe et al. 2001, 2007], but in different applications (e.g., 3D painting). There are just a few prior examples of bimanual 3D spatial interfaces for manipulating and navigating through volumetric scientific data (e.g., [Coffey et al. 2012c; Laha and Bowman 2013]), and none of these specifically support comparative visualization.

Bento Box also extends research on button overloading in VR (e.g., [Zeleznik et al. 2002; **?**])by providing an example of how this can be usefully employed for data visualization tasks rather than 3D modeling. This is achieved via a state machine that uses the current context defined by the positioning of the hands relative to the body, each other, and/or virtual content to decide how to interpret each button press.

3.3 Bento Box: Concept, Visual Layout & Interface

This section presents the concept, visual layout, and specific interactive techniques that make up Bento Box. Some figures refer to the cardiac application mentioned earlier as examples, but we defer a detailed description of that application to section 3.4.

We designed Bento Box to run on multiple VR environments that are popular for scientific visualization today, and demonstrate that the current implementation runs on both a high-end 4-wall Cave, which is useful to facilitate discussion with an engineering design group, and a low-cost HTC Vive personal VR display. The minimum required VR hardware is as follows. The technique relies upon a perspective-tracked, stereoscopic VR display and requires input from two 6 degree-of-freedom tracked input devices (i.e., VR wands), one held in each hand. Each device must have two buttons, one primary and one secondary that report separate *button_down* and *button_up* events. These input requirements are a subset of what is available via the hardware for the current-generation HTC Vive. In the 4-wall Cave environment we have used, users hold two tracked 3D pen-like devices [Jackson and Keefe 2016] to provide the simple button input together with 6-DOF tracking. Our implementation uses the MinVR toolkit [Jackson 2017] to facilitate deploying the application to these multiple platforms. The technique is designed to be operated by one user, but in practice engineering teams like to stand together and work as a group, looking over an operator's shoulder in the Cave environment pictured in Figure 3.1.

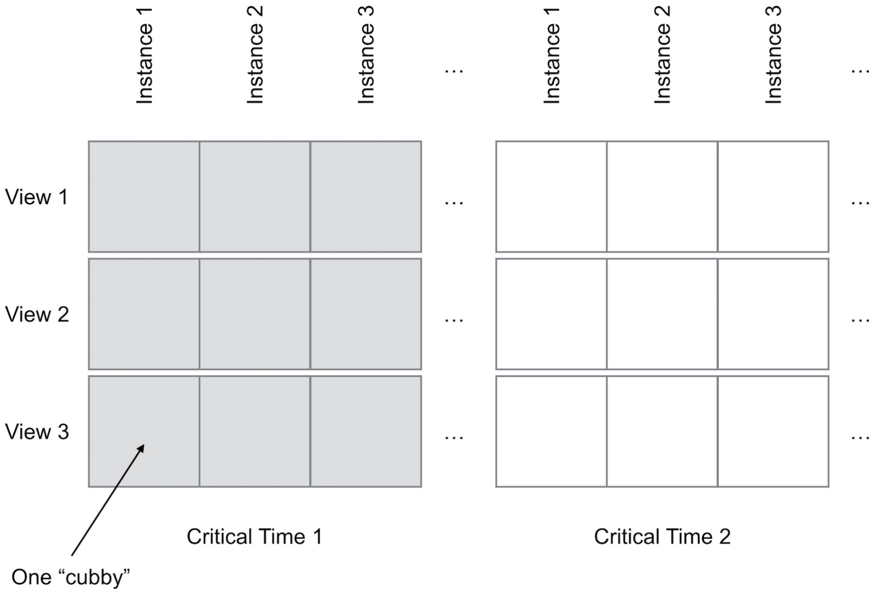


Figure 3.2: The Bento Box widget arranges multiple views of volume data within a grid of "cubbies".

3.3.1 Concept and Visual Layout

When users enter the environment, they see the "Bento Box" widget, an arranged grid containing multiple views of volume data, floating in front of them as in Figure 3.1. A 3D cursor is also drawn at the location of each tracked hand.

As diagrammed in Figure 3.2, each row and column have a specific meaning. Each row has unique *view settings*, which primarily define a specific sub-volume (i.e., a volume-of-interest) to crop from the original data along with the viewing direction. Additional parameters, such as the particular stress fields or other visualization properties to include in the view may also be set on a per-row basis. Each column presents data for a single *data instance* (i.e., the results of one simulation run) from the ensemble. If the user wishes to compare multiple timesteps (i.e., times-of-interest), this entire row-column structure can be duplicated multiple times.

Using this layout, the concept behind and intended use of Bento Box is for users to interactively create a comparative volume visualization where each column can be thought of as something like a volumetric, visual "feature vector". The user's goal is first to select appropriate sub-volumes of interest within the data in order to build up a complete picture of the interesting variation in the data while, importantly, cropping out or deemphasizing regions that are less important. The data comparison task then becomes to compare these visual feature vectors. For each key feature (defined in a row), users make visual comparisons across the data instances in the ensemble (columns).

The key to creating a useful visual summary of the data is to explore and experiment: looking at the data from different angles, creating views of new sub-volumes, changing visualization parameters, navigating to different views of the widget, and adjusting the viewpoint used to render individual rows. We aim for users to naturally, through this exploratory and interactive process, arrive at a visualization that brings the most important sub-volumes to the forefront of the visual field while at the same time hiding or deemphasizing distracting or less scientifically relevant regions of the volume. Bento Box makes the *process* of exploring the data and creating the ensemble visualization fluid, natural, immediate, and iterative—users can perform the navigation and display management regularly and naturally during data analysis without even taking their gaze away from the dataset.

3.3.2 Zooming and Reframing the Widget

Viewing the entire Bento Box is like looking at an overview of the entire ensemble. Some major, high-level patterns are visible at this level, but the visualization within each individual "cubby" is too small to investigate in detail. This makes it important to be able to fluidly zoom in and out of the widget and otherwise reframe it on the display in order to see just the current portion of interest.

Since zooming and reframing is so critical to the user experience, Bento Box supports two complementary interaction techniques, each appropriate in a different context. The first is controlled by the input device held in the dominant hand (DH) and is appropriate to use when the Bento Box is displayed at a small scale (i.e., used as an overview). In this situation, a laser pointer is used as the virtual cursor for the DH and the user simply points this laser at the Bento Box to highlight a specific cubby to investigate in detail. A click and release of the DH's primary button triggers an animated transition that zooms the view into the selected cubby. Alternatively, by holding the primary button down and sweeping the laser across multiple cubbies, the user may paint a selection onto the widget, and the view will zoom to comfortably fit the bounds of this selection. In each case the BentoBox is scaled so that the front faces of the cubbies selected by the user fit within a 1 physical meter x 1 physical meter area centered in front of the user. To return to the default position and scale the user clicks and releases the same primary button while pointing the laser away from the widget. (When explaining the interface to users, we make this interaction easy to remember by telling them, “to go back, simply point the laser backwards over your shoulder”.)

The one situation where this pointing interface is not efficient is when the view is zoomed in and the user wishes to make a small change (e.g., pan to the right by one or two cubbies). It is inefficient to do this by zooming all the way out and then back in to nearly the same position. Thus, the interface includes an ability to grab onto and translate the world directly. This is done with the primary button on the input device held in the non-dominant hand (NDH). To avoid unnecessary translations and rotations, this motion is constrained to only translate within the plane of the widget. Scaling is also possible. While the NDH primary button is depressed and the translation mode is active, pressing and holding the DH primary button activates the scaling mode. The scale of the Bento Box is then adjusted in proportion to the distance between the two hands. This second mode is like a constrained version of early VR object-manipulation interfaces [Cutler et al. 1997; Mapes and Moshell 1995], which are also similar to modern 2D multi-touch interfaces (i.e., translate with 1 finger, scale with 2 fingers).

Note that the interface intentionally overloads the functionality of the DH primary button—it means different things in different contexts. This strategy has been used successfully in several other bimanual 3D user interfaces [Zeleznik et al. 2002; **?**]. In general, the interface follows a pattern of using context to infer user intent whenever possible. This allows a complex interface to be specified using only two buttons on each hand-held input device and helps to overcome both learnability and “fumbling in the dark” problems that often arise in 3D user interfaces that use controllers with many buttons. It is important to consider the *state* of the system when designing and implementing this type of interface. Thus, Figure 3.3 presents a detailed Finite State Machine for the Bento Box interaction. The virtual cursors drawn in the scene change to provide visual feedback (e.g., from a laser pointer to a picking sphere) when moving from state to state.

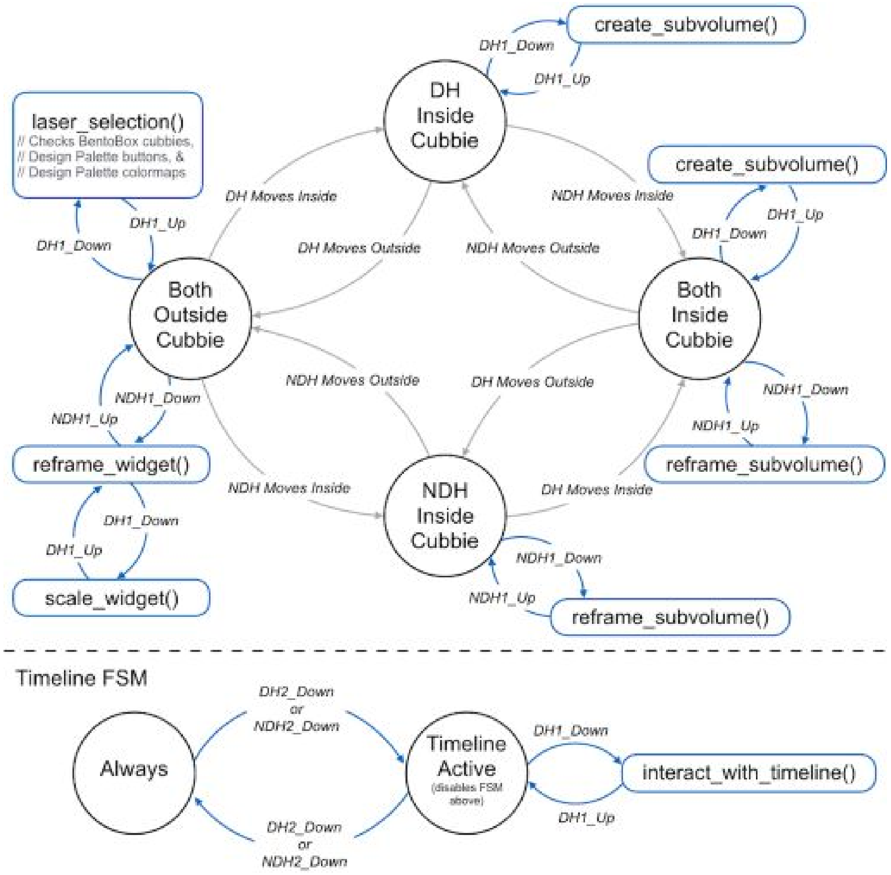


Figure 3.3: The 3D bimanual user interface is implemented as a finite state machine. There are four main states, and the system transitions between them based upon the positioning of the hands (DH = dominant hand, NDH = non-dominant hand) relative to the cubbies. As illustrated in the blue portions of the diagram, the actions triggered by the pressing the buttons on the VR wands held are different depending upon the context provided by the current state.

3.3.3 Creating and Reframing Sub-Volumes

As shown in Figure 3.3, a state transition is made when the DH cursor is moved within the bounds of the volume of a specific Bento Box cubby. Here, the concept is that the user is no longer in an overview mindset but rather in an inspection mindset. The cursor changes from a laser pointer to a small sphere to indicate this shift.

The most important operation in this state is to indicate a new (sub)-volume of interest and thereby add a new row of view settings to the Bento Box widget. This is

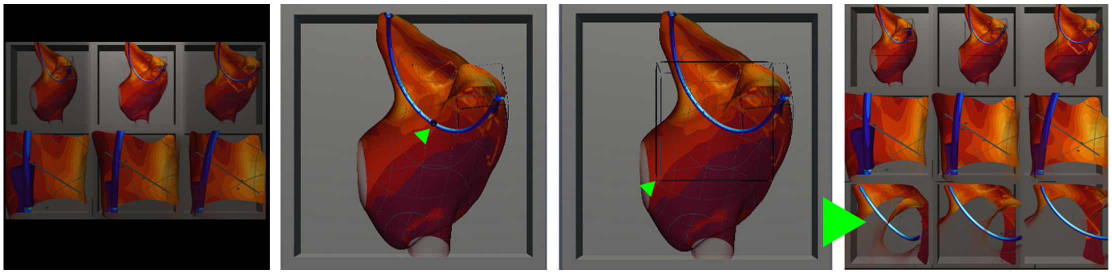


Figure 3.4: A new sub-volume of interest is created with an interactive selection. From left to right: (a) Assume the starting state is a Bento Box with two rows. (b) A click within any of the cubbies (in this case, one from the top row) using the primary button on the DH wand defines a center point for the selection (the black dot). (c) Dragging defines the size of the selection box to create. (d) After releasing the button, a new row of view settings is added to the bottom of the Bento Box.

done via a click and drag operation. The 3D location of the cursor at the moment the DH's primary button is clicked is used as the center of the volume of interest, and the extent of the volume is set interactively as the cursor is moved away from this center. It is critical to display interactive visual feedback during this operation (Figure 3.4) so that the user may size the volume appropriately relative to features observed in the data visualization. The selection operation is completed by releasing the primary button.

Creating the new sub-volume adds a new row to the Bento Box widget. All of the view settings are copied from the originating row with the exception that the transformation matrix used to draw the data within the widget is adjusted to exactly map the sub-volume displayed in the new row to that selected in the originating row. The exact transformation is described when discussing rendering in Section 3.3.6.

While the user is in this same inspection mindset, there is often also a need to view the data from a slightly different direction. Mirroring the use of the hands in the overview situation (DH = pointing/action, NDH = framing), this is accomplished with a grabbing operation controlled via the NDH. Again, the context provided by the positioning of the hand in space is used to distinguish a local grab that manipulates a sub-volume from the global operation that moves the entire Bento Box widget (see transitions in Figure 3.3).

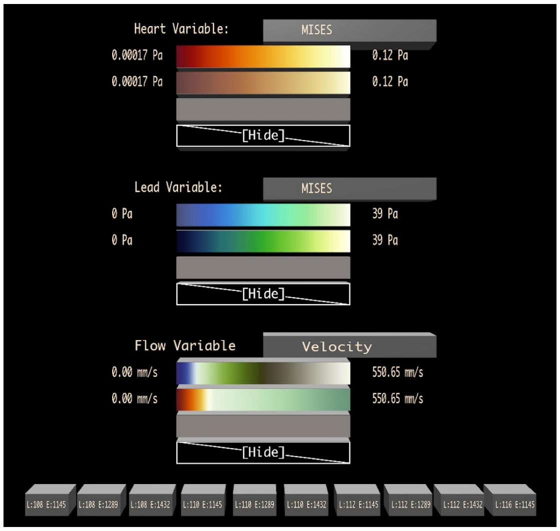


Figure 3.5: A visualization design palette is used to change the variable displayed on each major graphic element of the visualization (in this case: heart walls, lead, and flow) and to adjust the color map applied to the data. A set of possible color maps to apply to each variable is loaded during initialization, and the user may change the color map to display in each row of the Bento Box interactively by dragging one of these colormaps onto a specific row. Toggle buttons at the bottom of the palette control the visibility of specific data instances (the columns), labeled with instance parameters.

3.3.4 Changing the Visualization with the Design Palette

Figure 3.5 shows the visualization design palette, which is positioned to float in the air next to the user, coincident with the left wall of a Cave when running in a Cave display. The palette contains one section for designing the visuals used for each major graphical component of the visualization. Figure 3.5 shows an example from the cardiac application described in detail in Section 3.4. Here, there are three major graphical elements: the heart walls, the cardiac lead, and the flow path lines.

The heading for each section includes the name of the graphical element. Next to this is a button that can be selected with the virtual laser pointer attached to the dominant hand cursor and then pressed with a click on the primary button on the stylus. This cycles through a list of possible data fields that can be mapped onto the graphics (e.g., von Mises stress, Principle Stress, pressure).

Below this, the palette contains a set of color maps that can be used to present the data. With the laser pointer, a dragging operation is used to drag the colormap to a specific row of the Bento Box. While dragging, the laser cursor changes to include a colormap icon and rows of the Bento Box highlight as the laser passes over them to provide visual feedback. The use of dragging to control this operation is intentional, with one click and release, the user is able to specify both the specific color to apply (clicking while pointing at the design palette) and the row to apply it to (releasing while pointing at a Bento Box row). The last choice in each colormap list is a blank mapping ("Hide"), which removes the visual element from the scene completely.

Finally, an additional section at the bottom of the palette contains a set of toggle buttons, one per Bento Box column, to control the visibility of each data instance.

3.3.5 Using the Interactive Timeline

By default, each cubby in the Bento Box displays data from the same moment in time; however, additional time-points of interest (we call these *critical times*) can be added, extending the grid as diagrammed in Figure 3.2. Critical times are created and adjusted using an interactive timeline that is activated at a location in front of the Bento Box widget when the secondary button in either hand is clicked. The timeline acts as a modal widget, disabling the Primary Bento Box FSM shown above the dotted line in Figure 3.3 when the *Timeline Active* state is active. The timeline includes two virtual buttons that float in space and may be selected by the user's cursor. The first adds a new critical time to the display. This adds an indicator (color coded sphere) to the timeline at the correct time value and also adds the appropriate columns to the Bento Box. The critical times can be adjusted dynamically by grabbing onto the corresponding spheres on the timeline and moving them using the DH's primary button. While grabbing, the spheres can also be deleted by pulling them off the timeline by a distance of more than 0.3 meters and releasing. These interactions are demonstrated in the accompanying video.

3.3.6 Rendering Multiple Clipped Volumes

Bento Box requires rendering multiple views of multiple volumetric data instances. Each row is rendered using different view settings, which consist of: (1) an affine transformation matrix that transforms the raw volume data to a particular view of a sub-volume of interest, and (2) visual settings, such as the set of color maps and variables to be displayed. Since the view may require the data volume to be drawn at a scale that eclipses the size of its cubby, the rendering must be clipped to fit within the cubby.

For each cubby in the widget, the transformation matrix that maps the raw data to widget space, M_{D2W}, is composed of three parts and calculated as follows:

$$M_{D2W} = M_{V2W} * M_{C2V} * M_{D2C}. \tag{3.1}$$

Here, M_{D2C} (Data-to-Cube), transforms the bounding box of the raw volume data to fit within a unit cube centered at the origin. This transformation is specific to the data provided with each new application; the same matrix is used for every cubby in the widget.

M_{C2V} (Cube-to-View) transforms from the unit cube space to the view of the sub-volume selected by the user—this includes scale, rotation, and translation); the same matrix is used for each row in the widget. When the program starts, with only the top row of the Bento Box visible, M_{C2V} is set to the identity matrix. When a new row is created, the matrix to use for this new row is computed using the matrix from the originating row. The following equation is used, where where $S()$ constructs a scaling matrix, $T()$ constructs a translation matrix, and p and r define the center point and radius specified by the user (in View-Space coordinates).

$$M_{C2V_{newrow}} = S(1/r) * T(-p) * M_{C2V_{originalrow}} \tag{3.2}$$

Finally, M_{V2W} (View-to-Widget) transforms from the view space defined for each row to widget space by translating by a vector of the form $(cubby\_width*row, cubby\_height*col, 0)$ and scaling to match the current size of the widget. This is a simple translate-scale matrix, aligning the view to the specific cubby in which the data are to be rendered.

To clip each view to fit within its respective cubby, a clipping mesh is used. We chose a cube with rounded edges, but any convex shape of unit dimensions may be used. Before the view is drawn in a cubby, the back faces and front faces of the clipping mesh are rendered to two depth textures.

In our implementation, the graphics displayed for each data instance can be any 3D scene drawn using a traditional shader-based rasterization pipeline. There are two steps to adapt an existing rendering pipeline to work with Bento Box. First, the scene's model matrix $M_{original}$ should be modified using M_{D2W} before the Model-View matrix is computed.

$$M_{new} = M_{D2W} * M_{original} \tag{3.3}$$

Second, the fragment shader should be adapted to perform clipping to fit within two depth textures. For each fragment of the data scene, the fragment should be discarded if it falls nearer to the camera than a given Front depth texture, or discarded if it falls farther from the camera than a given Back depth texture.

3.4 Application and Results

The core Bento Box concept and technique described thus far, which we believe will generalize to other ensemble visualization problems, was inspired by the needs of a specific real-world data analysis problem, using an ensemble of fluid-structure-interaction (FSI) simulations to design improved medical devices. This section describes in detail the specific data management strategies developed for the application along with user feedback and quantitative performance measures from this first application of Bento Box.

3.4.1 Background: Cardiac Leads in the Right Atrium

Cardiac leads are the electrical cables that connect the heart to an artificial pacemaker device. The data visualized here come from a specific set of simulations designed to understand the impact of lead stiffness and lead length on the blood flow and stresses in the right atrium. The scenario is diagrammed in Figure 3.6. The goal of the study is to improve the underlying device technologies as well as procedures for implanting and extracting the devices. A 3×3 design was used for the initial study with three

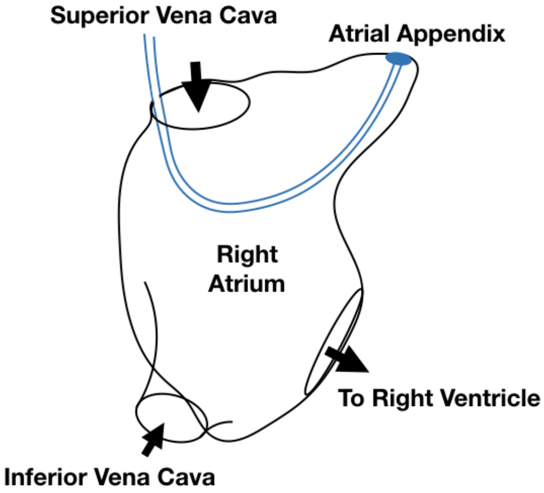


Figure 3.6: Diagram of the simulation scenario with a cardiac lead implanted in the right atrium of the heart.

lead lengths (108mm, 110mm, 112mm) and three lead stiffnesses (8N/mm, 9N/mm, 10N/mm corresponding to Young's Modules 1145.92MPa, 1289.16MPa, 1432.39MPa), resulting in 9 data instances. Later, one additional run (116mm, 8N/mm, 1145MPa) was added to the ensemble to understand the extreme case of extending the lead length as far as possible without touching the walls of the atrium.

The simulation extends Runesha et al.'s prior model [2016] and is built using the ABAQUS solver. The bounding geometry of the right atrium is a smoothed version of a real heart anatomy captured via CT scan, and the cardiac lead is modeled as a uniform wire entering the right atrium through the superior venae cavae and exiting through the tricuspid valve. Both the anatomy and the cardiac lead deform slightly over the course of a heartbeat. Each run consists of 800 timesteps for a 0.8 second heartbeat. Velocity and pressure within the volume along with stresses within the lead and along the walls of the atrium are saved at each timestep.

3.4.2 Sampling and Visualizing Solid Domain Data

As in most real-world ensemble visualization problems, some data management strategies are required in order to efficiently render many instances of the volumetric data. The approach described here is tailored to fluid-structure interaction (FSI) data and uses different strategies for solid domain data (Abaqus Implicit Solver) and fluid domain data (Abaqus CFD Solver). Since a VR rendering is required, the challenges of rendering large scale data cannot be solved with just an incremental loading or streaming approach, such as those used in recent 2D rendering contexts [Fisher et al. 2012; Glueck et al. 2014]. Here, there is also a 3D computer graphics rendering problem where the data to be rendered at each frame are simply too large to fit within graphics card memory. The techniques described here are similar to those described previously in the literature for solid domain [Beneš and Kruis 2015; Lee and El-Tawil 2008; Liangyin et al. 2018] and fluid domain [Falk et al. 2016; Kuester et al. 2001; Sobel et al. 2004; Zhao et al. 2017] data visualization, but we believe this research provides the first example of extending and using both styles of visualization simultaneously to display *multiple instances* of FSI data in a head-tracked, stereo VR environment. Thus, we provide a detailed account as a case-study-level contribution.

The solid domain describes properties of deformable meshes like displacement and stress and defines the physical structure of the volumetric solid with a mesh. The element and node properties of the simulation are used to generate a triangulated mesh that can be rendered. Triangles are constructed from the supplied primitives and passed to the GPU. Each instance uses its own index and vertex array since the hearts deform as a result of the simulation.

Since the solid data for all instances, variables, and time steps are too large to fit onto the GPU, optimizing the GPU memory and update speed becomes a significant challenge. We solve this by minimizing both the memory footprint on the GPU and size of data streamed onto the GPU at any time. In addition, we use CPU memory caching strategies to avoid disk access as much as possible. This involves only loading the variables and time steps that are actively being displayed into GPU memory. For example, if a user is only looking at displacement and stress for three instances, only these data are dynamically loaded on the GPU. However, depending on the CPU memory size, many time steps may be loaded into CPU memory, allowing for quick update if a user chooses to animate the instances.

The solid domain data are loaded into GPU memory when the user changes the displayed variable, the number of visible instances, or the critical times. Only the selected variable's values are loaded for each visible instance at the valid time steps, keeping the memory load footprint as low as possible. Each solid domain variable (e.g., wall stress) is assigned its own GPU buffer with size equal to the number of critical times multiplied by the number of FEA nodes and then by the number of components (1 float for scalar values, 3 for vector values). Although for perceptual reasons, our default is to view multiple critical times in static juxtaposed views, the visualizations can be animated by simply swapping data every frame, and this can be done automatically or interactively using the timeline widget described earlier.

3.4.3 Sampling and Visualizing Fluid Domain Data

The raw fluid domain data are large, but these data are visualized using particles, so a significant data reduction can be achieved by converting the raw data into pathlines in a precomputation step. Our implementation uses an accelerated cell location technique, similar to a CellTree [Garth and Joy 2010] data structure, to precompute path lines directly from the unstructured grid data using a Fourth Order Runga-Kutta integration method. A random seeding strategy is used; other seedings, such as targeted seeding in areas of high vortical structures, would also work.

The specific path lines to draw within each cubby are determined based upon the time and view. Since path lines inherently encode time; the display for a given critical time is determined by cropping each line to the portion that lies between the current critical time and a few moments before (in order to create a streaklet effect).

Bento Box requires rendering to be performed at multiple scales, and this raises an interesting challenge for drawing path lines. Recall that the path lines are just a visual representation for the underlying flow field, so when drawing them with a streaklet geometry, it makes sense to define the size of that geometry (its radius) in cubby-space units, not data units. Another way of thinking of this is that when zooming in, viewers do not wish to see a giant streaklet geometry, rather they want to see a more intricate visual rendering of the flow at that zoomed in scale. To accomplish this, both the size and density of the streaklets must be defined in cubby-space units.

Empirically, we determined that drawing about 2000 path lines per cubby provides the right balance for density—enough lines to provide detail to understand the flow and not so many that occlusion is a problem. This is a simple constant in our algorithm, and a different value may be easily incorporated to tune to technique for use with other datasest. What should not change from one application to another is the desire to maintain a constant visual density of path lines per cubby regardless of the scale of the data represented in that cubby.

To address this, the precomputed single set of randomly seeded path lines for each data instance is computed for the most-zoomed-in view expected. Then, any zoomed-out views that require fewer particles are drawn using just a subset of the precomputed paths; the size of the subset to draw increases as the view zooms into the data. The specific calculation is as follows. With the constant N as the application-specific desired visual density of paths per cubby, the number of particles to render, n, for a cubby displayed with scale factor, s, is

$$n(s) = N\left(\frac{s}{s_{external}}\right)^3. \tag{3.4}$$

The final constant in the equation, $s_{external}$, which is 4.0 for our data, is the scale at which the visualization transitions from an internal view of the flow data to an external one. With this formulation, n increases smoothly, randomly adding new path lines to the scene rendered and ultimately clipped into each cubby, as the view is zoomed in tighter.

The fluid domain rendering method makes it possible to visualize the flow at any scale and any critical time from the same pre-calculated data. Thus, changing the critical time does not use any additional GPU memory or require additional CPU-GPU memory updates. One path buffer array is stored on the GPU for each data instance in the display. The buffer is arranged according to a path index and each path has the same path length. The GPU also stores path value buffers for data variables such as velocity and pressure that may be used to color the particles.

The comet geometry is defined as an axis-aligned 3D mesh. Since each visualization will include thousands of these meshes, the mesh is rendered using instancing and deformed in a shader to fit within an appropriate start and end position along the path line. Our implementation uses a mesh with 72 triangles.

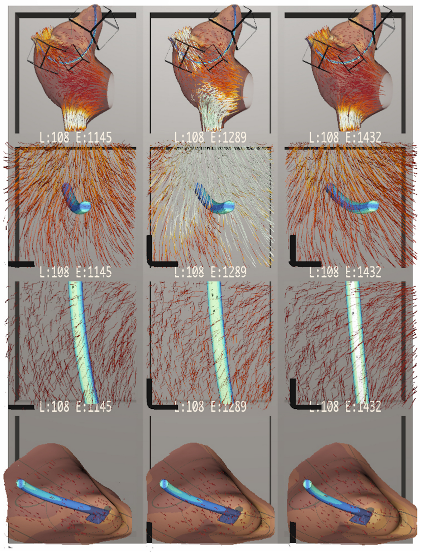


Figure 3.7: Bento Box arranged for a comparison of cardiac leads with three different stiffness parameters, increasing in stiffness from left to right. The top row shows an overview of the dataset. The middle two rows highlight stress on the lead itself, which appears to increase with stiffer leads. The bottom row zooms in on the attachment point of the lead in the atrial appendix, showing how in all cases the flow stagnates near the attachment point. Here, stress on the atrial walls also appears to increase with stiffer leads.

3.4.4 Expert User Evaluation and User Feedback

Our interdisciplinary collaborators have used Bento Box during several months of iterative development, most recently to analyze the scenarios highlighted in Figure 3.1 and Figure 3.7–Figure 3.8. The team includes two mechanical engineering researchers and four computational scientists who also confer regularly with cardiac surgeons and with engineers in the medical device industry. Several new insights about the data were able to be made. These observations come from multiple working sessions in the Cave, which is used regularly for collaborative data analysis by small groups of users. As mentioned earlier, the application also runs on the HTC Vive, and this has been a useful platform for portable demonstrations at international conferences and for school groups, industry, and university alumni; however, users have preferred the Cave for data analysis because it facilitates collaborative discussion.

Initial observations confirmed the expected spatial positioning of the leads, which

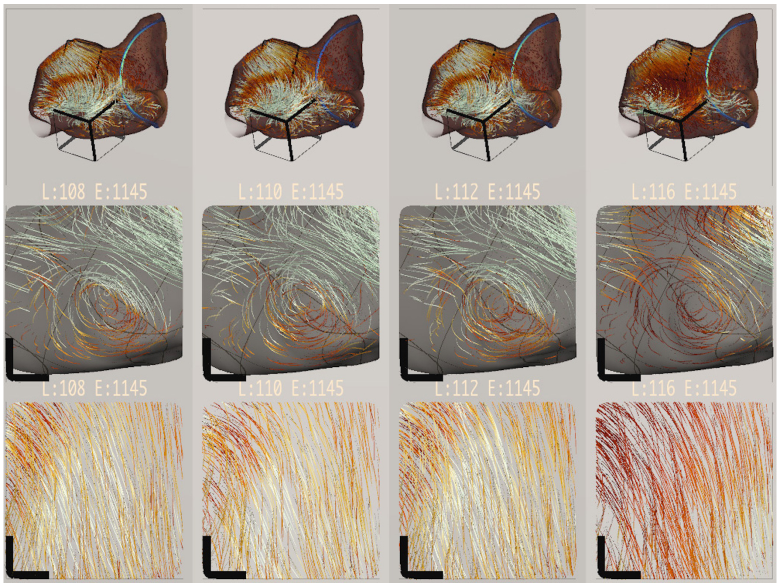


Figure 3.8: Bento Box arranged for a comparison of different length cardiac leads. Lead length increases from left to right. At this timestep in the simulation, the longest lead length creates a slower flow (i.e., darker red path lines).

was easily judged in VR. The engineers commented that the “drapings” for the leads in all data instances were appropriate. Looking at cross-sections of the lead by arranging sub-volumes that cut through the lead like a slicing plane, the engineers also confirmed that the internal stress pattern has a neutral axis, an expected pattern for a bending scenario like this one.

Figure 3.7 shows a comparison of data instances with leads of increasing stiffness from left to right. The second row shows the neutral axis stress pattern mentioned earlier. The sub-volume was rotated with a NDH gesture so that the front of the cubbies act as cutting planes, slicing into the finite element data. The third row shows a top-down perspective of the lead. Here, white indicates high stress. Engineers found that the highest stress on the lead occurs when the lead is at its stiffest, confirming their expectation. Conversely, the leftmost situation should have the most displacement. This was difficult to verify because the displacements are all quite subtle, and a suggestion was made to include a “motion magnifier” feature in future work to exaggerate any movement. The fourth row shows the volume of the right atrial appendage near the attachment point of the lead. This is a region where flow circulation and stagnation can occur and fibrosis develops. Here, engineers noticed that the stress on the atrial wall near the attachment point is higher (more yellow and less dark red) with stiffer leads.

Figure 3.8 shows a complementary comparison. Here, the focus is on different lead lengths, which increase from left to right. Some interesting variation in blood flow within the volume is visible. Engineers noticed that the longest lead produced flow patterns that appeared slower (darker red) and slightly out of phase with the other simulations (visible when scrubbing through time). This slowing trend is visible in the overview in the top row, but it can be seen even more clearly in the next two rows, which focus on a vortex and an outflow. The wall stress is hidden in these rows, and a neutral background is used to make it easier to read the color-coded variation. During the most recent data analysis session, this visual insight prompted several minutes of follow-on discussion, and the computational scientists hypothesized that the longer stiff lead might stretch the atrium wall, making the atrium bigger, and that the increased volume creates a slower overall flow pattern.

In terms of usability, the layout and interface controls made sense to users, who learned the controls within a first working session. One suggestion for improving the interface design was made. Users were sometimes confused when translating a sub-volume relative to a parent volume. Recall, from the discussion in Section 3.3 that this is supported via a grabbing gesture with the NDH. Users understood the rotational aspect of this grabbing, but had trouble with the translational aspect. One user told us that when she was translating she looked not at the sub-volume where her hand was located but at different row where she could see the location of the sub-volume displayed as an icon. When the user focuses on the sub-volume's icon, this breaks the metaphor of using the hand to grab onto and "move the data" and instead puts the user in the mindset of grabbing and "moving the selection box". Unfortunately, this does not work well in the interface because the translations will be the opposite of what is expected. The solution is not trivial. The widget is designed so that the boxes are arranged at fixed locations in space, so it breaks this design if the interface is switched to a mode of grabbing and moving the boxes. One option to explore in future work is to make the selection icons themselves objects that may be grabbed and moved with the hand. Then, if the user wishes to move the object, she simply finds a view where its icon is visible, grabs it and moves it. Alternatively, if she wishes to move the data, she grabs the data following the metaphor used in the current implementation.

3.4.5 Memory Usage and Rendering Performance

After processing the 39 GB of raw data, the amount of memory needed to accurately visualize the solid and fluid attributes is over 8 GB, exceeding a 4 GB GPU hardware limit on our 4-wall cave environment, a 2 processor Intel(R) Xeon(R) CPU E5–2640 @2.50GHz machine with two NVIDIA Quadro K5000 cards and 192 GB of RAM. Since this machine has more than 8 GB of RAM, it is possible to stream solid attribute data from memory into the GPU when needed. Streaming combined with the pathline sampling of the fluid, provides an extremely low memory footprint on the GPU, allowing us to visualize many instances and variables.

Using this application as a testbed, we also report some rendering performance measures, summarized in Figure 3.9. These timings were recorded on a 4 core processor Intel(R) CORE(TM) i7–7700HQ CPU @2.80GHz machine with 16 GB of RAM and a NVIDIA GeForce GTX 1070 graphics card, which was configured to drive an HTC Vive with a resolution of 2160×1200 pixels. The datasets are streamed into memory from a 128 GB M.2 PCIe SSD. The scatter plot in Figure 3.9 shows a systematic sampling of Bento Box configurations that are possible for this 10-instance data ensemble. All possible grid arrangements (10×1, 10×2, 10×3, 9×1, 9×2, 9×3, etc.) that result in a total of 40 cubbies or less were sampled. A cutoff of 40 was used since it is reasonable to assume that beyond this we reach a perceptual limitation in terms of what users can manage within the visual field. In fact, 20 cubbies is probably a better threshold. Since we found the rendering performance depends upon the zoom level, multiple samples at different scales were collected for grid arrangements that involved displaying subvolumes. In general, rendering speed decreases as the view is zoomed in, since this requires rendering more path lines to achieve the same visual density as at zoomed out views.

The trend is above 30 frames-per-second for Bento Box arrangements of about 20 cubbies or less, and is in the 40–50 frames-per-second range for typical arrangements, such as for the results pictured in Figure 3.7 and Figure 3.8. In some arrangements sampled, the frame rates drop below what we would consider a bare minimum for VR

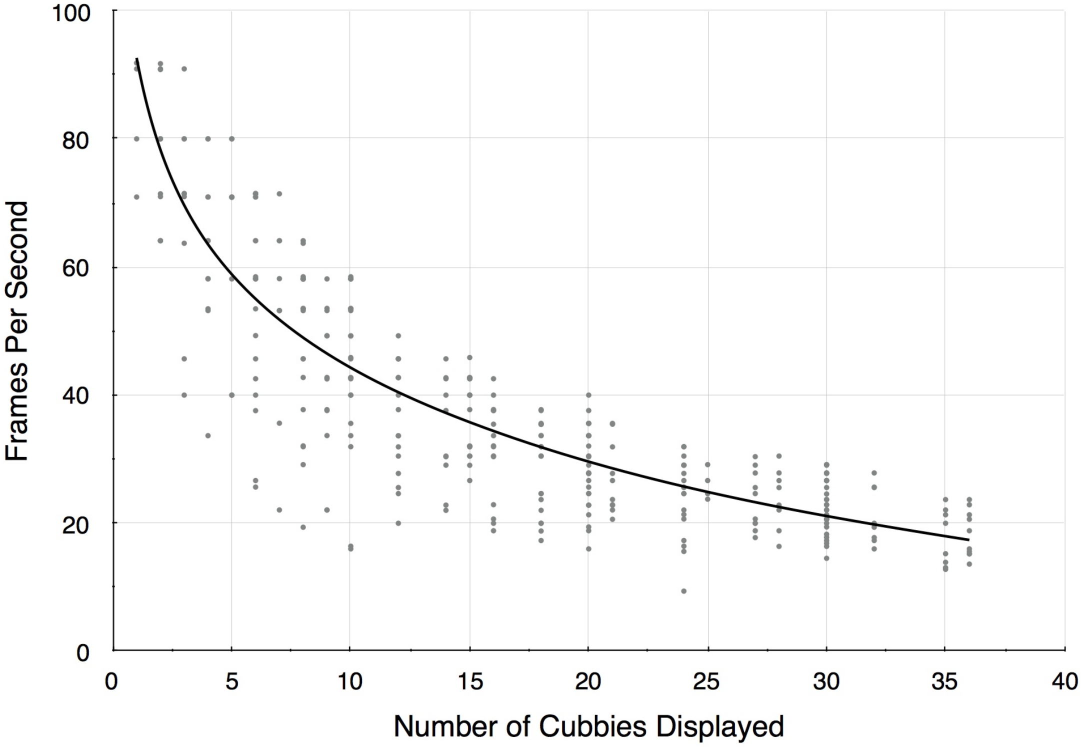


Figure 3.9: Rendering frame rates decrease as additional cubbies are added to the display. Since there are multiple ways to construct a Bento Box (e.g., there are 4 possible arrangements for 10 cubbies 10×1, 5×2, 2×5, 1×10), these results are for a systematic sampling of possible configurations. The trend line is a logarithmic fit ($R^2 = 0.76$).

environments (about 20 frames-per-second), but these cases are rare in practical use and have not detracted from analysis tasks using the system.

3.5 Discussion of Limitations and Future Work

There are two key limitations to the Bento Box technique that are worth reiterating. Although the concept, visual layout, user interface, and general rendering strategy can be applied to any 3D dataset, this only applies to cases where it is already possible to render the entire ensemble dataset. In fact, the ensemble needs to be able to be rendered multiple times per frame, in order to support multiple views at different scales.

Our application demonstrates that this is possible to accomplish with a realistic, real-world, scientifically relevant ensemble, but in practice it takes some work and requires thinking carefully about how to sample and render the data. Some simpler datasets (e.g., 3D geometry only with no flow data) would perhaps work without any rendering optimizations, but many of the ensembles that scientists are interested in studying today will likely need to be optimized for fast rendering. One goal of reporting this case study is to provide guidance on how to approach this task, at least for fluid-structure interaction data.

Another limitation is that Bento Box is not the right technique for large ensembles. We intentionally describe the technique as designed for ensembles on the order of 10 instances for two reasons. First, rendering is even a bigger challenge for larger ensembles. Second, perceptually, it asks too much of users to try to interpret a juxtaposed visualization that goes beyond 20–30 “cubbies”.

This leads us to the most important direction for future work. We see great potential to combine Bento Box with a workflow that includes filtering. In this way, large ensembles might be able to be interactively filtered down to sets of 5–10 most interesting instances, then these could be explored in VR using Bento Box. This might be enabled, for example, by adding a linked scatter plot visualization of a dimensionally reduced view of a large ensemble from which the user could select individual or groups of instances to add to the Bento Box. Perhaps this could happen at a central control panel within a virtual room with multiple Bento Boxes created based on specific filters arranged around the virtual space. The workflow could also be extended in the other direction, making it possible to dive deeper into individual volumetric data instances to query specific data values with a probe or place other interactive visualization widgets directly within the detailed visualizations inside each cubby to access details on demand.

Finally, we now know, since Bento Box helped us to analyze the ensemble, that the variation within the particular cardiac lead data ensemble developed as part of the research project is quite subtle, and we are curious to learn how Bento Box would work in other situations, such as an ensemble where the variation between data instances is drastic. It would also be interesting to use Bento Box to explore abstract data, like a 3D field of data glyphs. In this case, the rendering optimizations described for FSI data would not be necessary, but, assuming the data are dense enough that zooming in is required to do detailed analysis of sub-regions, we hypothesize that the core technique would be just as valuable as it is for medical volume data.

In the future, we plan to conduct additional evaluations of the technique, for example, it might be possible to design a formal user study to assess speed and accuracy in a search and comparison task conducted with Bento Box vs. a standard juxtaposed or interchangeable (over time) comparative visualization. This would be a significant undertaking, likely requiring generating a synthetic volumetric ensemble dataset with features that can be interpreted by non-expert users. It may also be possible to implement a baseline VR or desktop-based visualization that uses an alternative visual approach to comparison, recruit additional experts with knowledge of cardiac medical device engineering and fluid analysis, and then design and execute a formal insight-based evaluation [Saraiya et al. 2005] of the support Bento Box provides for interpreting the data used in our case study. More generally, Bento Box is a good example of a visualization tool that, in practice, is often used in a collaborative data analysis mode; it would be interesting to more formally assess the strengths and weaknesses of various VR platforms (e.g., head-worn display vs. Cave) for this type of analysis.

3.6 Conclusion

VR environments are already effective for visualizing simulation data with complex spatial relationships, such as those presented in this chapter, but only when visualizing a single data instance at a time. To make VR visualization useful for comparative visualization of a data ensemble, we conclude that new techniques for spatially arranging and cropping the data are necessary, since these help users focus attention on the most important 3D and 4D regions of comparison. Such an arrangement is only possible within VR with the aid of a tightly integrated 3D user interface. The Bento Box technique addresses both of these needs. Further, the application described here demonstrates that it is possible to use Bento Box to construct a visualization of a 10-instance, real-world, scientific data ensemble and provides early indication of the potential impact of visualizations in this style.

Bento Box is successful at providing an interactive visualization of complex 3D datasets. The careful selection of colormaps and the custom reframeable views enabled our collaborators to produce imagery that allowed discernment regarding their research questions. And the fluid bi-manual immersive interface brought a level of interaction that engaged our participants by making the data accessible in a way it hadn't been before.

However, we felt as though we hit a ceiling in this project in the extent to which it demonstrates the pillars of palpability. Some examples are as follows:

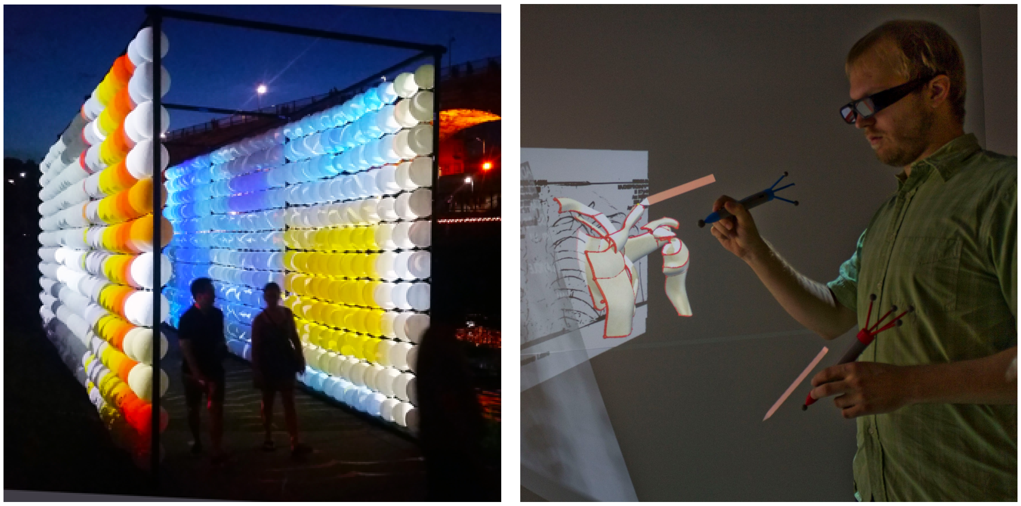
- **Discernible:** While artfully chosen, our colormaps only allowed us to show one scalar variable per surface per view, requiring the user to actively change which variables they were interested in viewing at any one time. It would be valuable to find ways to increase the visual bandwidth by being able to encode multiple scalar values on the same surface.
- **Discernible:** The computer graphics imagery in Bento Box is procedurally generated from the datasets, and this results in a classically "computer-generated" appearance. While this isn't necessarily a problem for focused analysis tasks, it does lack a strong visual connection with the real-world objects being modeled, and often requires a significant amount of explanation to new users. Our collaborating artist suggests the ability to specify flow glyphs and surface textures that more effectively communicate the type of data being displayed might prove to be a beneficial addition.
- **Accessible:** While we were ultimately able to include our artist's chosen colormaps, it was not a very fluid process, requiring us to email images of the colormaps back and forth, and re-launch the application to try a new colormap. Bento Box does not feature an interface for designing and iterating through different colormap concepts.
- **Accessible:** Since Bento Box was developed for a large VR Cave environment, we required our collaborators to either come and visit our research lab, or participate in remote screen-sharing sessions that stripped away all immersion. While we eventually ported the application to an expensive performance laptop for use with the HTC Vive, it is still challenging to travel with the system and share the Bento Box experience with new users.

Our exploration of state-of-the-art scientific visualization through Bento Box leads us to step back and consider how the creative design process might be more natively supported in our software. The interest and insight of our artist proved extremely valuable, but there was a notable barrier to her involvement in the design of Bento Box. By more deeply integrating an artist's creative process with the design of expressive scientific visualizations, we expect the visualizations to be more palpable, making the data more discernible and more accessible.

Chapter 4

Studies in Accessible Design

The artist's approach to designing immersive experiences



(a) Weather Report (b) Lift-Off

Figure 4.1: Two immersive scientific visualization projects with artist-oriented design processes. (a) Weather Report, an interactive outdoor art installation which provides an evocative comparison of objective climate data and participant-driven subjective memories. (b) Lift-Off, a existing artistic hybrid 2D/3D sketching system, applied to visualizing medical data.

4.0.1 Introduction

To investigate the potential of designing data visualizations with artists in-the-loop, we look look at two projects that were developed in conjunction with artistic processes. By taking the time to understand how artists iteratively create experiences, and how those experiences may be focused on scientific problems, we hope to learn how we would develop visualization design workflows to better involve artists. This chapter presents two case-studies, Weather Report and Lift-Off, which each provide insight into different artistic processes applied to scientific data.

Weather Report [Keefe et al. 2018; Swackhamer et al. 2017] is the result of an collaboration between architects and computer scientists working together to design and construct a piece of interactive data-driven sculpture art. Our work is in response to a 2015 call-to-artists for interactive art installations that explored the idea of climate crisis to be displayed over one night at an annual art festival held in Minneapolis, Minnesota. The interdisciplinary team worked together over the course of 6 months to discuss how we might fuse interactive data visualization with architectural design. Through an inter-mixed series of brainstorming sessions, concept sketches, site visits, critiques, and prototypes, we created Weather Report. This chapter, in part, recaps our collaboration on Weather Report as an example of a design process accessible to artists, resulting in a visualization which was profoundly accessible to the public. At the end of this chapter, we will review what makes this interactive installation a palpable experience.

Lift-Off [Jackson and Keefe 2016] is a pre-existing VR Cave-based modeling application from our research lab allowing artists to sketch 2D drawings in the physical world and import these into the virtual world, using the drawings as starting points for fluidly constructing 3D sculptures with hand-held tracked styluses. This tool caught the interest of our interdisciplinary team of medical device engineers and computer scientists who had also collaborated on Bento Box (Chapter 3). We were specifically interested in exploring how a design tool like Lift-Off could fit into the workflows of designing medical devices and communicating pre-operation information to patients. We approached and published these applications as case-studies in how workflows designed with artists in mind could facilitate the creative and expository tasks of medical experts [Johnson et al. 2016].

4.1 Study 1: Weather Report

[1] As citizens struggle to understand the scientific language and real-world impacts of climate change and scientists struggle to recapture trust in data and scientific processes, how might artists, designers, scientists, and other thought leaders contribute to local public discussions of climate, and even science and data more broadly? This question, along with our own human desires to create, brought our design collective of architects, landscape architects, and computer scientists together to develop an interactive installation for the Northern Spark dusk-to-dawn, multidisciplinary arts festival in Minneapolis, MN, USA.

Climate Chaos / Climate Rising

The Northern Spark arts festival is held each year in the Twin Cities and has grown to attract tens of thousands of people who view performances, explore temporary installations in the streets and along the riverfront, and gather for one night to experience art as a community until the sun rises. In 2016, artists were asked in an open call to interpret the theme of “Climate Chaos / Climate Rising” through “the creation and presentation of art in the public sphere, focusing on innovative uses of technology, old and new, to imagine new interactions between audience, artwork and place, explore expanded possibilities for civic engagement, and encourage pluralistic community” [NorthernLights.mn 2016].

Art, Design, Science, & Technology

Our design collective, MINN_LAB, approached this challenge by fusing our plurality of design voices into a vision of a data-driven nighttime experience with local weather data. We identified common themes around the experience we wished to create: *making climate personal* and *connecting objective scientific data to subjective human experiences.* Then, over the course of months of interdisciplinary design and discussion, we developed, interpreted, and revised these themes through the lenses of our different technical

---

[1] This section is based on work published in IEEE Computer Graphics and Applications [Keefe et al. 2018].

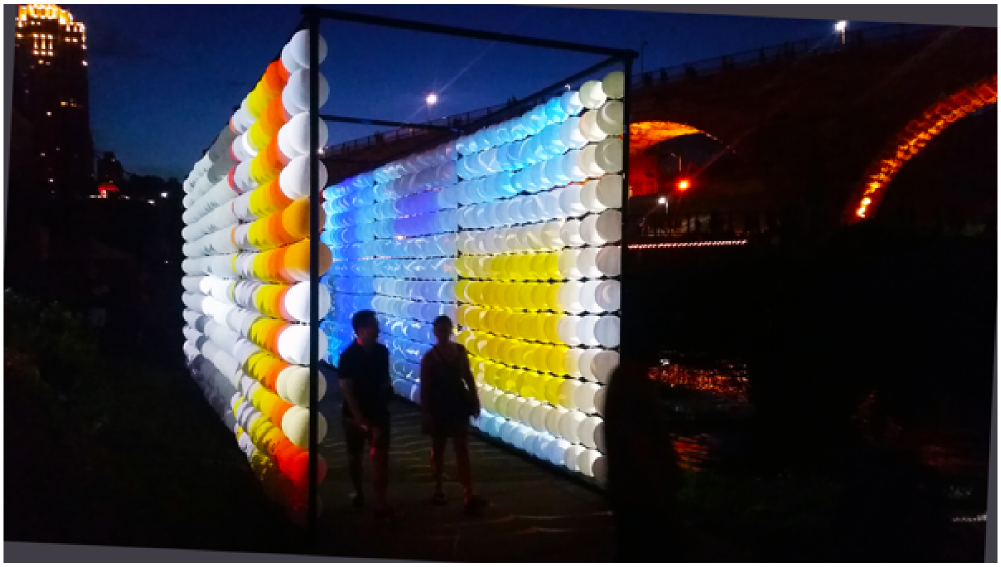


Figure 4.2: Visitors walk through a tunnel of animated "balloon pixels" that depict 4.5 decades of local weather data constructed from both objective measurements (right wall in this view) and the subjective memories of visitors (left wall).

research interests: for the architects, *understanding the role of data in the built environment*, and for the computer scientists, *imagining a future of data visualization based on embodied experiences and physicality.* The result is Weather Report (Figure 4.2), a human-scale, visualization tunnel that contrasts 4.5 decades of objective local weather data with the subjective weather-related memories of visitors.

4.1.1 Weather Report Concept

Weather Report uses local, human experiences with weather as an entry point for discussing the difference between objective weather data and subjective interpretations and memories. Although the data visualized are on the human timescale (45 years of objective data and memories), the objective-subjective comparisons the piece asks visitors to make speak to the broader question of how the earth's climate has shifted over a much larger period of time, how this is measured and interpreted today, and how rigorous scientific processes differ from everyday discussions of weather.

Scientists often struggle to explain the objective basis for climate change, what it means to the average citizen, and the grand timescale on which it operates. In contrast, our friends and family have no trouble at all explaining everyday human experiences with weather and extrapolating from these: *On the day you were born, there was an amazing blizzard, it took me four hours to shovel the car out of the driveway; that used to happen all the time, but we don't have storms like that anymore.* Or, just as common, *we have never had a winter with so many violent storms in a row; grandpa had to buy a generator because the power kept going off; this year is the worst ever.* Weather Report asks visitors, which of these weather memories is true? How do human experiences and memories compare with objective data? How do human timescale data points relate to the much larger climate timescale, and how do scientists objectively measure those data?

Getting Physical

Inspired by the role weather balloons play in data collection, we came to view the balloon as a simple, physical, relatable manifestation of the scientific process. Each weather balloon provides a small data-driven contribution to the larger picture of the science. Reinterpreting this in the context of an experiential display of data, we reasoned that balloons could function as physical "pixels", changing appearance in response to individual data readings and collectively presenting a broad picture of the scientific data.

The balloon-as-pixel concept could have many interpretations, and this fueled a rich, several month-long period of sketching and ideation within the team. Drawing upon the architectural tradition, the site for the installation was critical in the design. Mill City Ruins Park is located along the Mississippi River, near the Stone Arch Bridge in Minneapolis. One of the prominent features of the park is a walking path that follows the river. Thus, we were inspired to create a walk-though experience – something viewers could experience and touch as they traversed the path.

The method for illuminating or otherwise adjusting the balloon pixels in response to data also required design. We experimented with balloons on strings with motors and internal LEDs but found that the best visual results could be achieved by projecting colored light onto the balloons. The light is transmitted through the balloon, which creates the effect of a glowing orb.

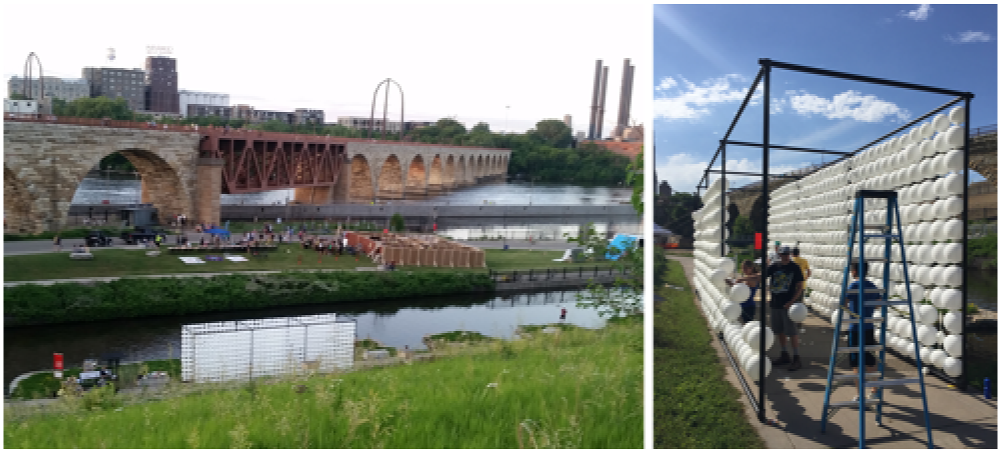


Figure 4.3: A tunnel of balloon pixels along the walking path in Mill Ruins Park, Minneapolis, MN.

Contrasting Views of Weather

The final design arranges the balloons into two walls that form a tunnel around the walking path (Figure 4.3). Each wall is made from a 12x36 grid of balloon pixels. This mirrored walls arrangement establishes a strong physical framework for visual and body-centric comparison of the data visualized on the two sides of the path, and this serves as the basis for addressing the key theme of the work, contrasting objective and subjective views of data. One view is assigned to each wall.

The wall next to the river visualizes objective local weather data, recorded at the US Weather Station KMSP located at the Minneapolis - St. Paul International Airport. A dataset was created based upon hourly temperature, wind speed, and precipitation (rain and snow) readings from 12AM on January 1st 1960 to 7:00 AM on June 11th 2016. In all, the objective data include approximately 4.5 decades of hourly readings for these four variables.

The wall next to the grassy hill visualizes subjective data. Visitors populate this dataset based on their subjective, weather-related memories, which are entered into the Weather Report database using an interactive multi-touch kiosk positioned along the walking path near the entrance to the tunnel. The subjective dataset contains the same variables as the objective dataset but is much sparser. Over the course of the night, visitors fill in this subjective weather record, and the visualization interpolates between

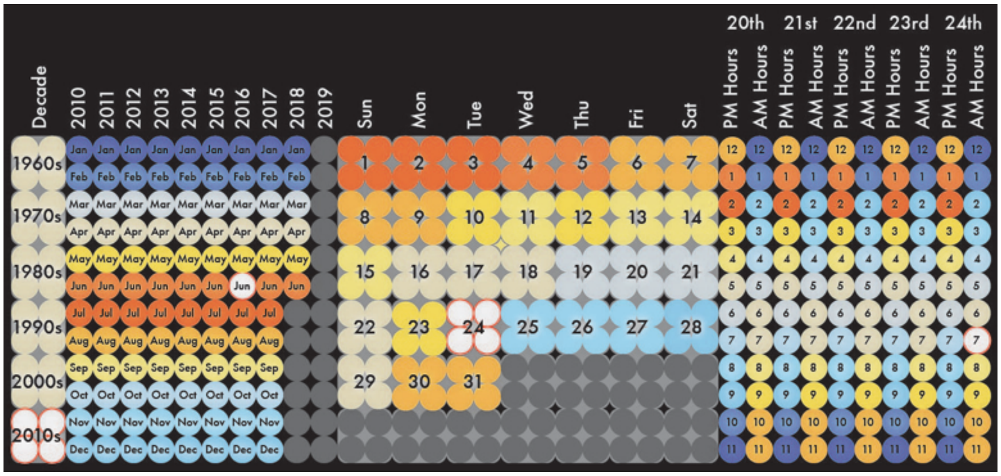


Figure 4.4: The data-to-visual mapping for each wall uses a hierarchical arrangement for time, decades are displayed in the two leftmost columns, with the current decade highlighted, followed by months of the year with the current month highlighted, followed by days of the month, and so on. The display animates to display the entire six decades of data over the course of the night.

each data point to fill in the gaps.

4.1.2 Visualization and Interaction Design

The visualization is designed be experienced. The visual mapping is accurate, making creative use of hierarchy, time-window averaging, and animation to (over time) present all of the multi-decade, multi-variable data using only the available 12x36 pixels; however, the primary goal is to evoke a sense of being present with the data rather than an accurate scientific analysis. The same mapping is used for each wall so that at any point the color of a balloon on the objective wall may be compared to its partner on the subjective wall.

Mapping Temperature Data to Balloon Walls

The balloon pixels are organized into a hierarchical time grid, as illustrated in Figure 4.4 where each balloon visualizes data within a certain time window. In the leftmost two columns, the time windows are the largest, covering entire decades of data. Then, the time windows get progressively smaller moving from left to right.

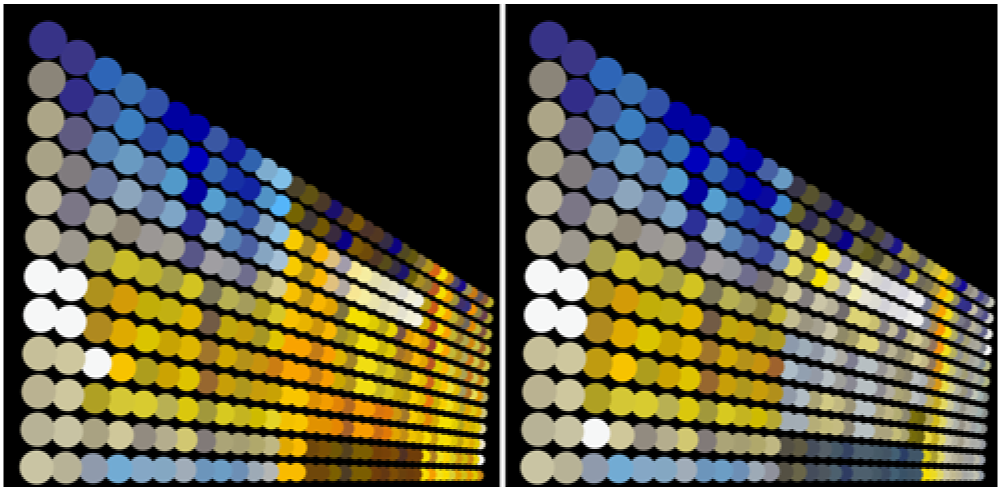


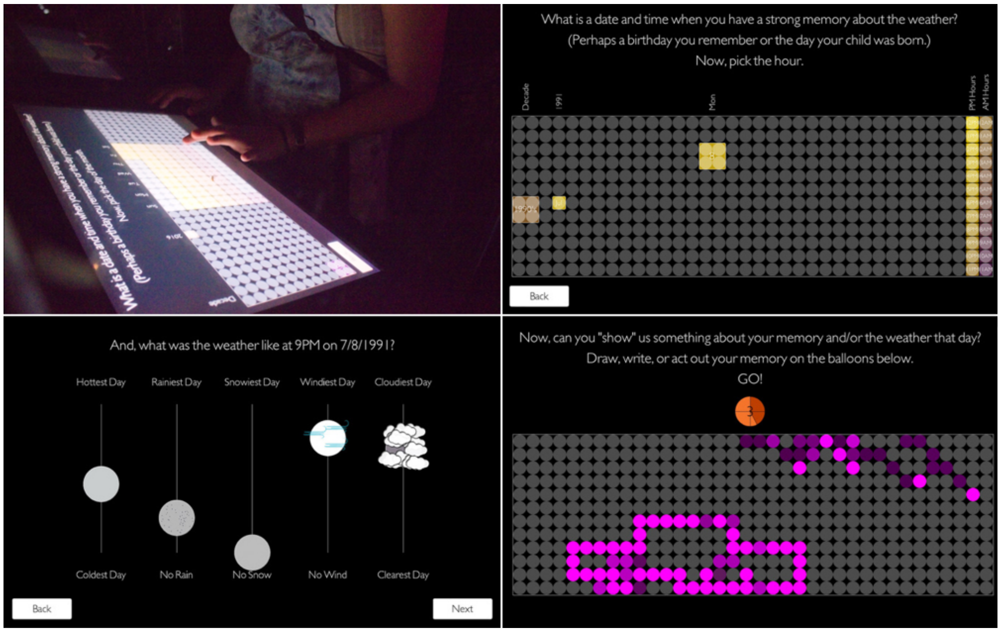
Figure 4.5: Two frames from an animated blue rain effect temporarily superimposed over the objective weather wall.

Since the display is animated, the visualization maintains a "current time", which is a single hour within the dataset. The balloons corresponding to this hour are always drawn with a white highlight. Note that there are always several such balloons due to the hierarchical arrangement – the current hour is highlighted on the far right, and the current day, month, and decade that contain this hour are also highlighted within the other sections of the display. Figure 4.4 shows the highlighting for 7am on Tuesday, June 24th, 2016 as an example.

The color of each balloon is defined using a warm to cool color map based on temperature data. Temperatures are averaged within the time windows, so that a balloon corresponding to a day-long time window displays the average temperature for that day, balloons corresponding to a decade show the average temperature for that decade, and so on.

Animated Effects for Secondary Data Variables

In addition to the primary variable of temperature, secondary data variables (rain, snow, wind speed, and cloud cover) are also included in the visualization, but these use a different visual mapping. The secondary variables are treated as discrete weather "events". An event is present at a given time in the dataset if the rain, snow, wind



What is a date and time when you have a strong memory about the weather?
(Perhaps a birthday you remember or the day your child was born.)
Now, pick the hour.

And, what was the weather like at 9PM on 7/8/1991?

Now, can you "show" us something about your memory and/or the weather that day?
Draw, write, or act out your memory on the balloons below.
GO!

Figure 4.6: Visitors enter weather memories using a three-stage, multi-touch interface on a kiosk.

speed, or cloud cover exceed a threshold. When the animation's "current time" reaches an hour that includes one or more weather events, an animated weather effect is applied to the visualization.

These animated effects are special in that they do not operate on a single pixel but rather "take over" the entire display. Whenever a weather event is encountered in the data, a three-second effect is added to the animation queue. For a rain event, blue pixels stream down the wall. Snow events create slower, gentler white pixels wafting down the wall. Cloud-cover events tint the top of the wall light gray, and wind events cause gray particles to fly across the wall from one side to the other. The effects are rendered as a semi-transparent overlay as shown in Figure 4.5.

The Subjective Weather Record

Visitors to Weather Report create the subjective weather dataset themselves by entering weather-related memories using a multi-touch kiosk located along the walking path, near the entrance to the tunnel. The user interface is pictured in Figure 5 and is designed

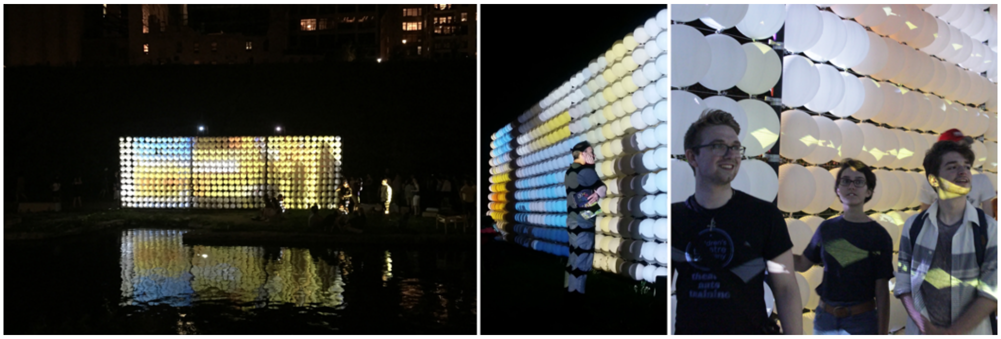


Figure 4.7: Thousands of visitors experienced Weather Report at Northern Spark

to also serve as a teaching tool – visitors learn the data mapping illustrated in Figure 3 by using the balloon wall graphic as the interface for entering the date of their weather memory (Figure 4.6, upper-right). A specific hour for the memory is entered by selecting the decade, then year, then month, then day, and finally hour. Then, the temperature, rain, snow, wind speed, and cloud cover are entered using sliders with extremes labeled in relative, not numeric terms (e.g., the hottest day ever vs. the coldest day ever) (Figure 4.6, lower-left). Finally, visitors act out their weather memory, using the multi-touch screen to create an animation (Figure 4.6, lower-right).

Similar to the animated effects for secondary data variables, these weather memory animations are replayed as graphic overlays immediately after entering the memory and again whenever the animation reaches the time associated with the memory.

Illuminating the Balloon Pixels

The balloons are lit from outside the tunnel by four 5000 lumen short-throw projectors. The projectors are fed real-time images of brightly colored ellipses (as in Figure 4.3) that align with the physical balloons.

In the Mill Ruins site, the tunnel is tightly pressed from both sides by a river and a steep hill (with sometimes less than 6 feet of margin), so the projectors cannot be positioned to allow for a perpendicular throw – instead, the riverside projectors must be placed off-axis, beyond each end of the tunnel (as seen on the right edge of Figure 6, left), and the hillside projectors are placed above the tunnel along the incline (the two lights seen above the wall in Figure 4.7, left).

A custom image-warping process is used to accommodate these extreme angles. The hillside projectors each illuminate about half of the subjective wall, while the riverside projectors each reach the entire objective wall from grazing angles. The custom software makes it possible to interactively specify anchor points using keyboard commands on-site to carefully align the projected ellipses with the balloons.

4.1.3 Observations, Surprises, and Reflections

Thousands of visitors experienced Weather Report at Northern Spark (Figure 4.7), and more than 200 contributed memories to the subjective weather record. The earliest entry was for Dec. 13, 1961. There were 22 entries for the Halloween Blizzard of 1991.

The biggest surprise was in how viewers reacted to the projection. As designers, we treated the projection as simply a behind-the-curtain technology – the technology that just happened to provide the best method of illuminating the balloons. To our surprise, the projection beams became interactive play areas, places for visitors to dance, pose for pictures with data covering their bodies, and cast shadows on the data that could be seen from across the river and around the festival. Visually, this added another layer of human connection and embodied movement to the data-driven visuals.

Interdisciplinary collaborative design processes are both rewarding and challenging. Our experience and results reflect months of discussion, much of it devoted to learning to speak the language of each other's disciplines. The design also reflects push and pull and compromise. The whole team had to work within the constraints of an outdoor, dusk-to-dawn festival and each discipline had to stretch to accommodate the interests and expertise of the other. This process led to a unique result, one that no individual on the team would have created. It also led to new thinking that broadens and refuels our primary disciplines.

This is one of the powerful recurrent themes in the work highlighted in this study. This team continued to collaborate as an interdisciplinary collective for another similar project, and even as we returned to our primary disciplines we brought new knowledge with us that impacts our future work. Architects have learned to become a little bit more like computer scientists and now have a foundation for further explorations of data and science in the built environment. Computer scientists have learned to become a little bit more like architects and now have a foundation for further explorations of embodied experiences and physicality in data visualization. Computer graphics and visualization researchers will connect this work to the emerging research theme of "data physicalization" [Jansen et al. 2015]. This is a topic where the computer science community has much to learn from architects, designers, and artists. Collaborations like this one make it possible to explore concepts of data in physical space at a scale that is simply impossible in a traditional computer science context.

4.2 Study 2: Lift-Off for Medicine

[2] When designing a visualization for accessibility, it is hard to beat the accessibility of the physical world, both for construction and presentation. In Weather Report, we saw how accessible the design of an experience is when artists can work in the processes with which they are most familiar, and how accessible an experience itself is when it can be touched and walked around. In this section, we take our insights from Weather Report and apply them to a virtual design space for scientific tasks.

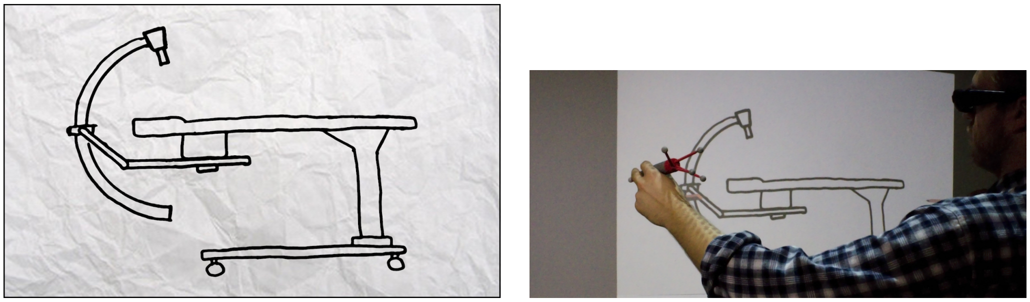
4.2.1 Introduction

Society has benefited greatly from recent advances in medical imaging and data-driven design of medical devices. However, the ability to design, analyze, interpret, and communicate about medical data remains a challenge. We believe that by combining virtual reality interfaces with creative data-intensive work-flows, new immersive analytic tools can address this challenge.

In this section, we present two application case studies showing how hybrid 2D/3D sketch-based interfaces in VR, specifically interfaces from the recent Lift-Off immersive modeling system [Jackson and Keefe 2016], can be used to support immersive analytics in the medical domain. Lift-Off (Figure 4.8) allows a user to position 2D imagery within a 3D virtual environment shown in a VR Cave. The user can lift 2D contours out of the imagery into 3D space to create a wire-frame network of rails. Surfaces can then be swept along the rails creating a 3D model with precise control.

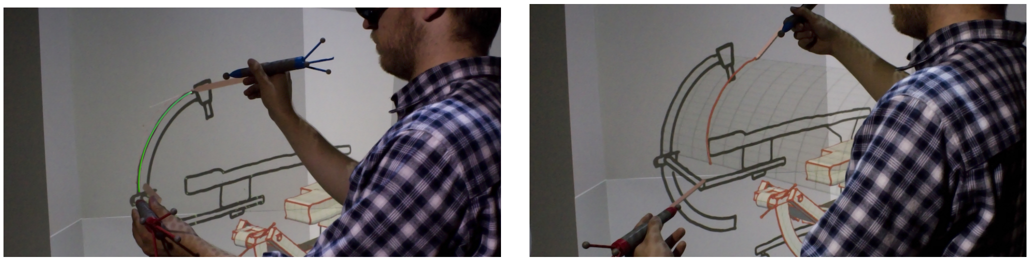
---

[2] This section is based on work published in the Proceedings of the 2016 ACM Companion on Interactive Surfaces and Spaces [Johnson et al. 2016].



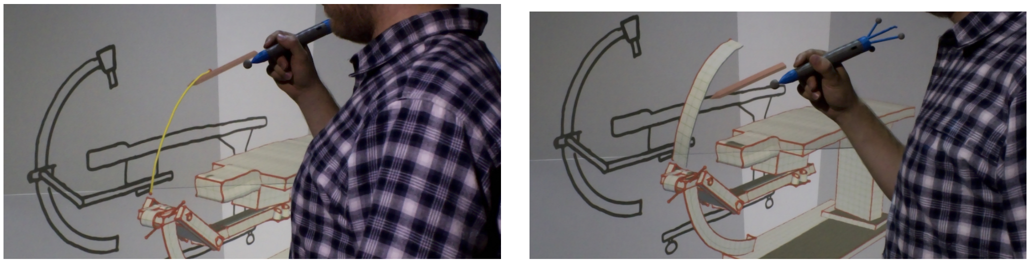
(a) Start with sketches on paper.

(b) Scan and place in 3D space.



(c) Select edges of interest.

(d) Lift these curves into 3D.



(e) Adjust depth along the curve.

(f) Sweep surfaces along curves.

Figure 4.8: The process of designing from a sketch in Lift-Off.

The first application explores how immersive 3D modeling based on 2D X-ray imaging can be used to analyze and explain bone fractures to patients before receiving care. The second application explores how creating a VR scale model of a medical device from early 2D sketches might facilitate a new style of engineering design for medical devices. Both case studies were developed by our interdisciplinary team of medical device engineers, computer scientists, and a physician. Based on these experiences, we argue that hybrid 2D/3D interfaces that support new creative output in addition to visualization are critical for the development of immersive visual analytic tools. (A video demonstrating this system can be found at: https://youtu.be/taTwUWYh4jw)

Our main contributions are:

- A case study of combining a hybrid 2D/3D sketch-based interface with medical imaging data to support surgical intervention for bone fractures and increase physician/patient communication.
- A case study of combining a hybrid 2D/3D sketch-based interface with hand-drawn 2D sketches on paper to support designing new medical devices in an immersive spatial context.
- A discussion of challenges for the medical domain as a fertile application area for immersive analytics.

4.2.2 Related Work

Medical Analytics and Visualization in VR

Medicine has always been an important driving application area for immersive visualization and analytics. Immersive environments have been created to help scientists and physicians analyze computational simulations of blood flow [Van Dam et al. 2000], understand the structure of the brain [Zhang et al. 2004], and plan and train for surgical procedures [Satava 1993]. Now, as immersive technologies have become dramatically more accessible and affordable, there is an opportunity to do much more. Our first case study explores the potential of using VR not as an analytical tool for only the most complex neuroscience surgical intervention, but rather for the everyday task of a physician working together with a patient, helping the patient (or medical resident) to reason in 3D about why a procedure is necessary. In the future, we also envision a dramatically increased role for interaction relative to prior work in immersive medical analytics. Our second case study explores the potential of immersive environments that are not just for interactive visualization of existing datasets but also for creating data (i.e., designing, 3D modeling).

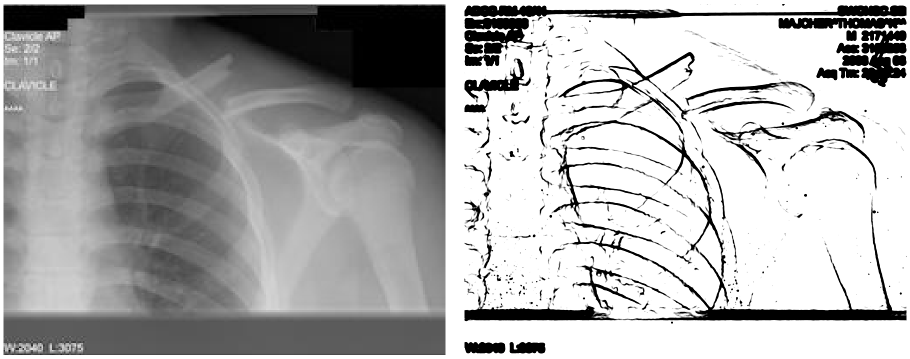
Sketch-Based Modeling and Annotation in VR

Early work in sketch-based modeling tools for VR includes the 3DM system [Butterworth et al. 1992] which supported the creation of 3D surfaces by sweeping a tracked six degrees-of-freedom stylus through space. This was followed by Holosketch [Deering 1995] and many others. In general, 3D sketching in VR has been embraced for its immediacy and the ease with which even novice users are able to create complex models. However, user feedback shows that unconstrained 3D input is difficult to control. The Lift-Off modeling system [Jackson and Keefe 2016] avoids much of this issue by introducing constraints from hand-drawn 2D sketches (Figure 4.8). In addition to modeling, freehand sketch-based systems have been used to annotate and describe scientific data. Keefe et al. explored how to prototype scientific visualizations using sketched 3D input [2008]. Nowinski et al. studied surgeons' ability to annotate vasculature structures in VR for surgical planning [1997]. We believe incorporating engaging, full-body, gestural interfaces in these styles into immersive data visualizations is the thing that is needed to create a next generation of successful immersive data analytics tools.

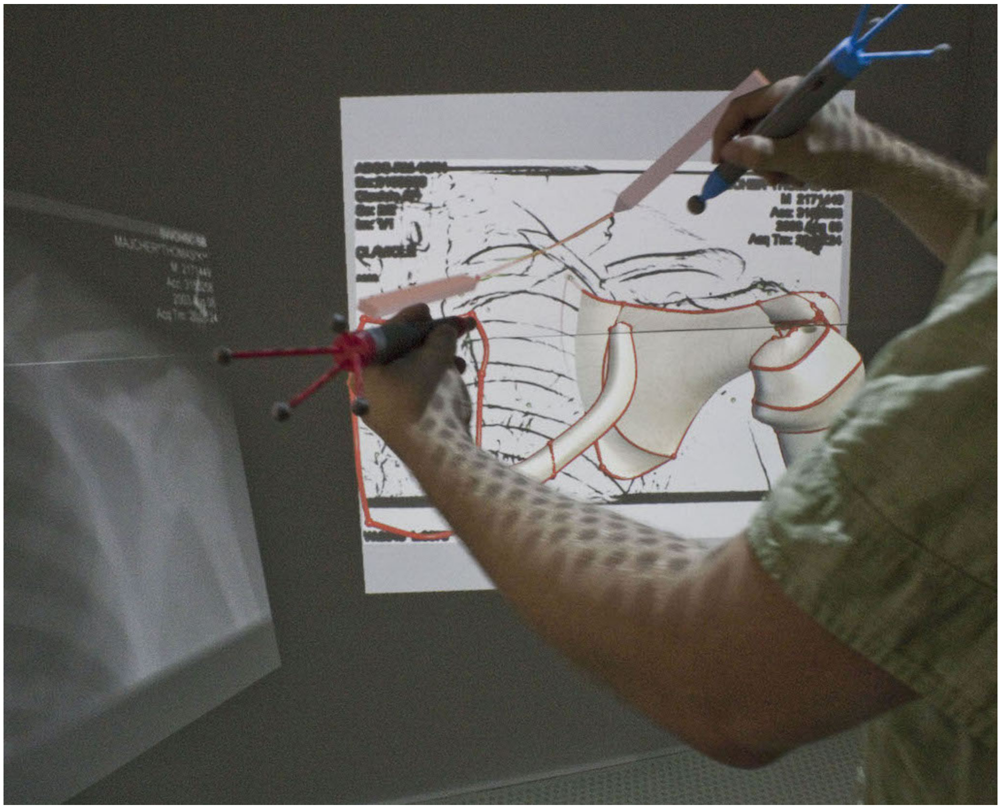
4.2.3 Application 1: Annotation of Medical Data

Physicians need to quickly and reliably translate medical information to patients and their family members, and this often involves translating concepts captured in 2D imagery or other data to the 3D context of a patients body.

Prior work with Lift-Off has focused on translating artist-defined 2D line-sketches into 3D virtual models for architecture and sculpture art, but we reasoned that a similar hybrid 2D/3D interface might also be useful for physicians to facilitate patients' comprehension of medical imaging data.



(a) Starting with a standard X-ray image (b) Applying an edge-detection filter



(c) Users place these data images as slides in 3D space and then “Lift-Off” curves to construct 3D models and annotations

Figure 4.9: 2D medical data can be converted to line images for use with Lift-Off. Data credit: X-ray image used through Creative Commons from Majorkev on Wikipedia https://creativecommons.org/licenses/by/3.0/.)

Methods and Results

To test this potential, we developed an example use case based on the concept of translating the 2D medical data captured in a patient's X-ray to a 3D context that might be more easily understandable to the patient. We considered the case of a broken clavicle bone. Figure 4.9(a) shows the original digital X-ray image, and Figure 4.9(b) shows the same image after applying the simple edge-detection filter built into the Lift-Off tool. Together these images are placed (like floating slides) in 3D space within VR, and they serve as the data context for 3D illustrations and annotations. Figure 4.9(c) shows how construction lines were selected from the edge data and pulled out from the image to a user-defined depth to construct the anatomy relevant to the discussion.

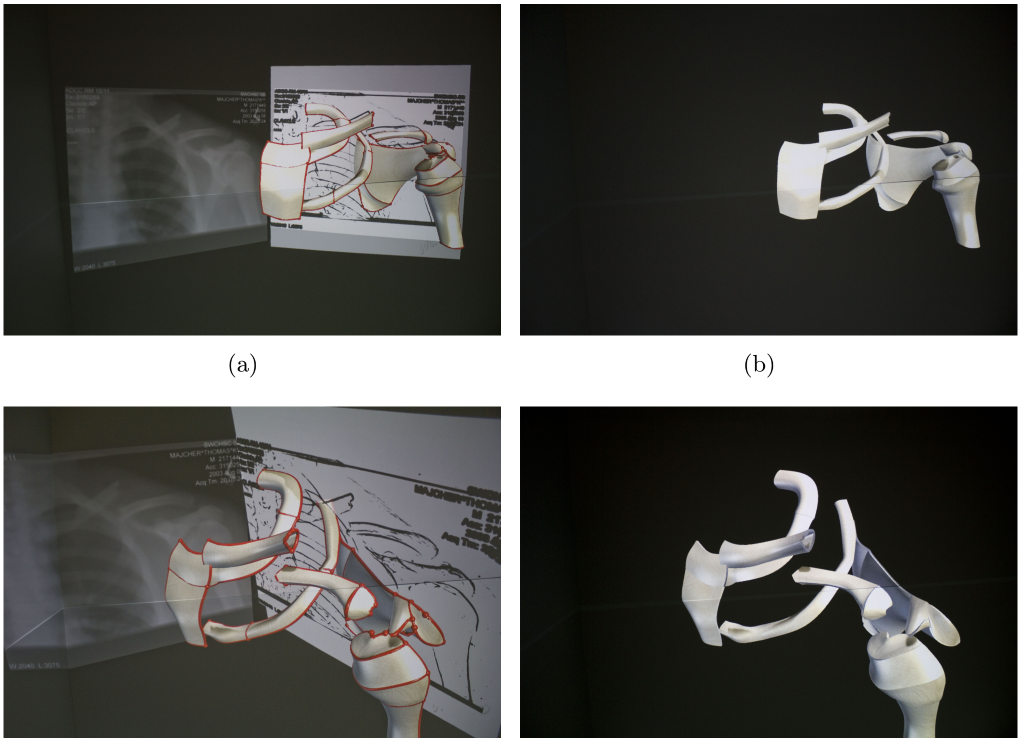
To illustrate this specific medical example, we lifted out geometry for a section of the sternum, the first and second rib, the broken clavicle, a section of the scapula, and a section of the humerus. Several views of the 3D scene this process generated are shown in Figure 4.10. This particular model took an experienced user (the first author) about 1 hour to construct in Lift-Off.

Observations and Feedback

Our interdisciplinary team generated several observations from this experience, and we recorded feedback from the physician on the team as he engaged with the tool to demonstrate how he might use it to better communicate with a patient or resident (Figure 4.11).

One of our first observations was a surprise. The physician's first step was not to discuss the break in the bone specifically; rather, he began decisively sketching areas of concern near the fracture such as blood vessels, explaining that the bone fragments could cause further internal damage if not treated, a danger he indicated was critically important for the patient to understand in order to pursue appropriate treatment. Interestingly, this is precisely the type of 3D context that is not visible on the 2D X-ray image; here, it was only made visible with the new ability to sketch in 3D around the data-driven context provided by the X-ray image positioned in space.

We also noted the feedback that this clavicle fracture case is perhaps a bit too simplistic to convey the real need for a tool in this style. For example, a more compelling



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

Figure 4.10: A complete 3D model of the broken clavicle from two angles. Shown both with ((a), (c)) and without ((b), (d)) the design scaffolding.

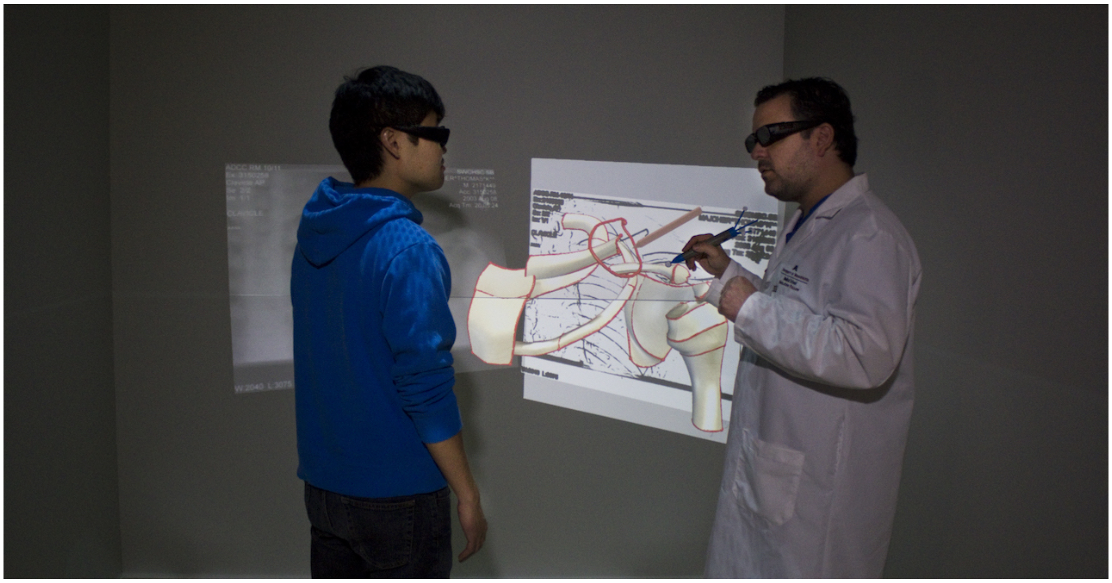


Figure 4.11: Using immersive data-driven 3D annotations to explain treatment options.

example of the need for communication might involve broken ribs in elderly patients – here, it can be difficult to convey to the patient the need for careful supervision in a setting where the patient can remain still, breathing deeply for a consistent period of time so as to avoid developing pneumonia. He went on to demonstrate how he would describe an even more complicated condition, pancreatic cancer, using freehand 3D sketching to create a diagram in the air showing the pancreas and surrounding organs and adding arrows as annotations while describing the treatment process.

The primary conclusion from our observations is that these situations would benefit from a virtual 3D white-board that allows physicians to draw as they talk in the context of both 2D images and 3D anatomy. This would enable physicians to annotate the 3D reconstruction, describing procedural and securement methods and identifying implant locations. The current case study succeeded to a degree in making this possible. However, there were also some shortcomings. The physician noted that while physicians would likely not want to take the time to actually model the contextual bones and anatomy in practice, they would make extensive use of the ability to sketch 3D diagrams in the context of generic 3D anatomy and 2D medical images. We interpret this as a need to extend the data visualization supported in our prototype, currently limited to just 2D medical imagery, to 3D visualizations that include surface and perhaps even volumetric models for organs and other structures. Imagine, for example, performing the type of annotation and modeling described here within the context of a state-of-the-art 3D visualization of neural fiber tracks visualized in VR.

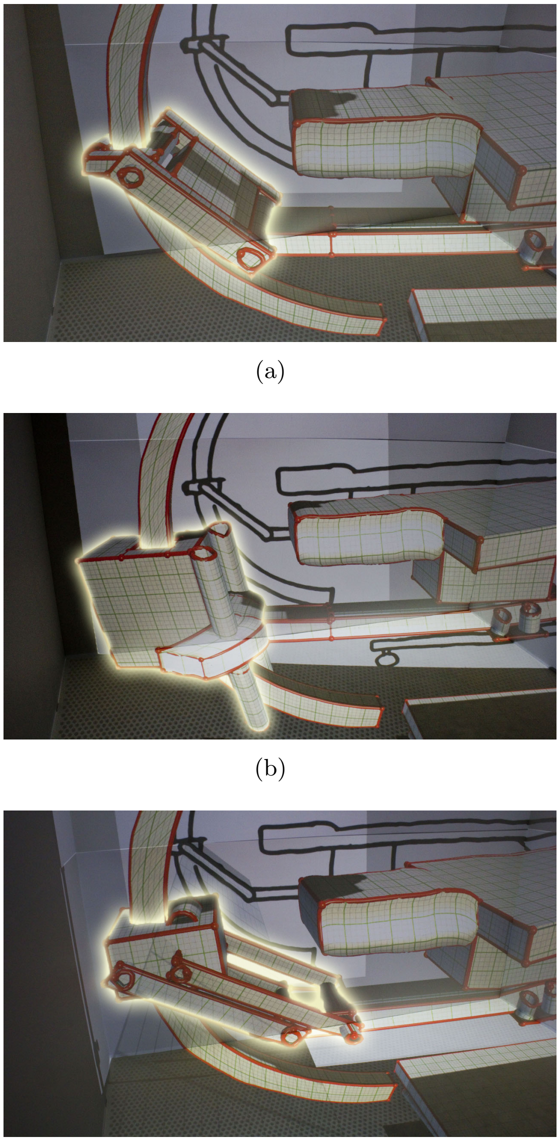
4.2.4 Application 2: Immersive Medical Device Design

Medical device design is a collaborative process. Designers need to conceptualize ideas for new medical devices and relay those concepts to engineers for further refinement and prototyping. This often starts on paper, but because the 3D complexity of these models can be so high, the process then often quickly moves to computer-aided design (CAD). One of the limitations of this quick transition to CAD tools is that once we move to a CAD model, with precise geometry, constraints, etc., we lose much of the quick, creative, and exploratory benefits of sketching. On the other hand, it is clear that traditional 2D sketching can only go so far, particularly when we consider designing medical devices with complex geometries that might be inserted within the human anatomy.

Methods and Results

To demonstrate how Lift-Off can be used in the medical device development process, we made several sketches on paper to capture ideas for a table-based robotic surgery device that our interdisciplinary team had been discussing for several weeks as part of another project. Working from one of the sketches, we created a full 3D model (i.e., a virtual 3D sketch) of the device in VR (Figure 4.8).

In this specific example, a successful design must be able to robotically control the position and orientation of a laser relative to the patient's head while the patient lies on an operating table. Precise positioning is required, and the design must also address several spatial constraints, such as room for the medical staff to work, room to transfer the patient on and off the table, and room for the surrounding equipment in the operating room. The design can be modified at real-life scale inside VR, and variations can be explored. For example, Figure 4.12 shows three variations for the hinging mechanism. All three of these designs were created based on the same 2D sketch by using free-hand 3D sketching and brainstorming. The first model took an



(a)

(b)

(c)

Figure 4.12: Three variants of the robotic mechanism, sketched in the 3D immersive environment.

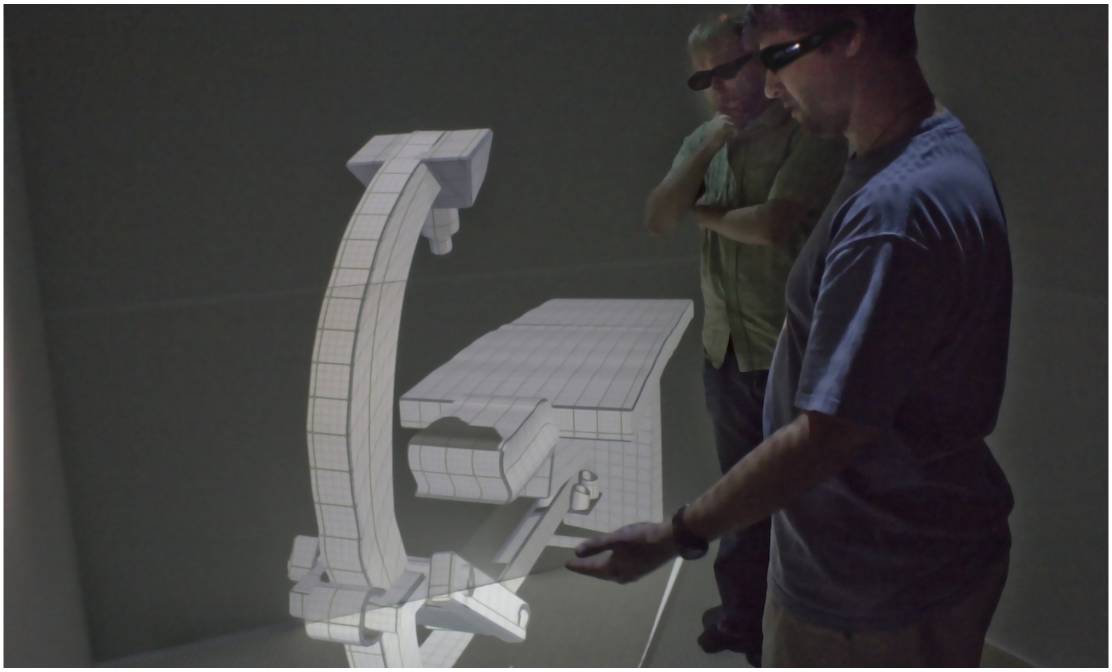


Figure 4.13: Critique of the robotic surgery device sketches can occur directly in the immersive environment.

experienced user (the first author) about 1 hour to construct, and each variation took an additional 15 minutes.

Observations and Feedback

The medical device engineers on our team evaluated this process and provided insights as the process moved from conference room ideation and sketching on whiteboards and paper to VR, where critique focused on the 1:1 scale 3D model shown in Figure 4.13.

An observation we made was how the approach to discussing the device changed when one of our medical device engineers encountered the virtual prototype. First, upon entering the immersive environment, the engineer appeared energized as compared to the conference room discussion. When another member of our team noticed that our system did not actually allow the different parts to move relative to each other, he took the opportunity to draw arrows in the air around the virtual prototype to indicate the degrees of rotational freedom of the multiple moving parts. Interestingly, these 3D arrows showed movement that would be difficult to visualize together on a 2D sketch.

When asked to reflect on the design variations of a particular hinging mechanism (Figure 4.12), one engineer commented on possible implementations involving four-bar linkages and sliding systems with clamps that could be powered by hydraulics. Before the sketch and virtual prototype were developed, this hinging mechanism had not even been discussed. While in the cave, members of our team stood at different positions around the virtual prototype to see how the various components might get in the way of potential surgical operations.

Our conclusion from our observations is that putting people in a 3D space with a life-size and life-like virtual prototype facilitates discussion and allows for considerations of implementation and spatial constraints. All design decisions can be made with attention to scale and functionality within the surrounding environment. Without a hybrid 2D/3D VR tool, these sorts of discussions and considerations would be inhibited until a physical prototype could be produced. However, there are also shortcomings to using Lift-Off over physically prototyping, such as the inability to physically interact with the virtual prototype or to move individual parts.

Although this particular case study focused on a large-scale medical device, our team members also work regularly with smaller scale and implantable devices (e.g., replacement heart valves, cardiac leads, drug delivery systems). This is an area where we think the interfaces described here can be combined with data-rich immersive visualizations to create immersive analytics systems that are powerful. Imagine, for example, the style of collaborative 3D design and sketching described here coupled with the style of immersive visualizations of blood flow through the heart mentioned earlier when discussing related work.

4.3 Conclusions

4.3.1 Weather Report

Our work on Weather Report resulted in a deeply accessible visualization, engaging over a thousand visitors in one night. Some people only walked between the walls on their way by, some took the time to contribute their own data, and some stayed for long while just taking in the display (Figure 4.14). Not only could participants reach

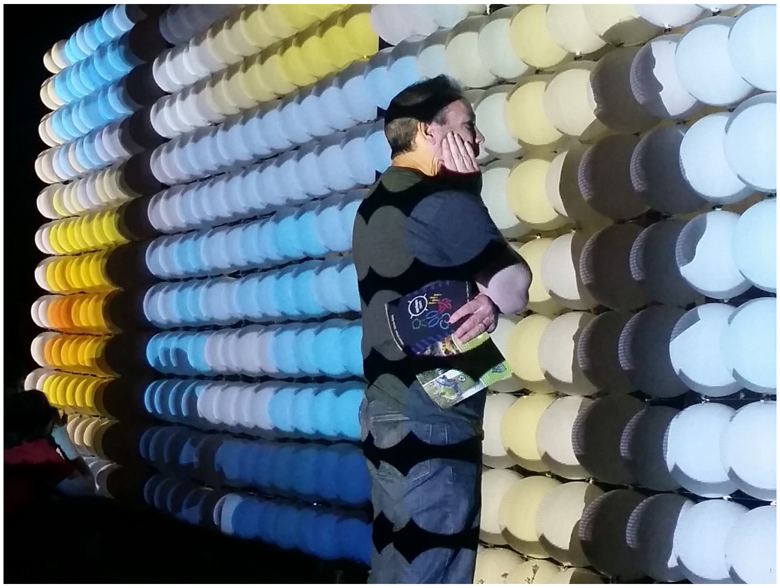


Figure 4.14: Weather Report drew in visitors to not only experience a visualization of climate data, but reconsider their subjective understanding of climate as it relates to objective information.

out and touch the balloons making up the walls, but they could use their fingers to literally draw their contributions before seeing their input displayed larger-than-life on the subjective wall.

In terms of palpability, this visualization is one of the most accessible works in this dissertation. This can be credited to the involvement of our team of architects, who have dedicated their careers to designing physical spaces that engage the people who inhabit them. We can also consider the high level of access our designers had to the visualization design process. There were minimal barriers to the architects exercising their design training during the development of Weather Report, simply because the result was undeniably an architectural accomplishment. Contrast this with what would have happened if we were primarily designing a piece of visualization software. While our artistic collaborators may have had valuable insight for us, and may have been helpful in the brainstorming, it would not have been in their domain of expertise and in the end we as software engineers would need have needed to re-synthesize our interpretation of their ideas anyways. This motivates us to find ways in other projects to enable our non-programming collaborators to play to their strengths, and incorporate design elements that are native to their domain.

However, Weather Report shows its weakness when it comes to discernibility. While it successfully encodes hourly temperature data across 6 decades and engagingly represents hundreds of user-contributed memories, the actual insights that can be made from the display are far more conceptual than they are analytical. The exact encoding is lengthy to explain, and easy to forget when confronted with the glowing array of animated balloons. This is a combination of the inherently low-resolution of the display, and the single channel of visual encoding – color – like we discussed regarding Bento Box in the previous chapter. While we don't consider this to be a failure for Weather Report itself, we do see this as an area for improvement as we continue to explore palpable visualizations. In Chapter 5, we will reconsider ways more data can be expressed at once in a scientific visualization of many variables.

4.3.2 Lift-Off

Moving onward in our investigation of creative workflows, we wanted to consider how to further leverage physical creative processes in our interactive software. In the first study, we performed most of our design work in the architectural domain, taking full advantage of their creative design workflows. In Lift-Off, we examined a software-based approach to creative design, while still integrating a physical creative process of sketching ideas on paper.

In the this second study, we explored the potential of using a 2D/3D hybrid user interface to sketch on top of medical imaging data, not just as a way to plan a high-end surgery (although this could certainly be useful) but as a way to even perform the much more common task of explaining a medical procedure to a patient. Similarly, we explored the potential of using a VR system as a 3D sketchpad for medical device engineers to create new device prototypes in immersive environments. We believe the two most successful elements of these case studies are: (1) the freedom to sketch and create new 3D data with reference to a data visualization rather than simply providing interactive techniques to explore a preexisting dataset, and (2) the ability to link one's prior experience with 2D data (or even physical sketches on paper prepared outside of VR) to the new immersive 3D environments in which people will work in the future.

With Lift-Off, we can demonstrate that a design process beginning in the physical world makes design tasks accessible not only to artists, but to anyone who has spent their life wielding a pen and paper for sketching. And when a design is displayed as an immersive environment that supports natural annotation, it also provides accessibility for new collaborative insights. When considered along side Weather Report, these two works reinforce our idea that integrating a physical creative process increases the over-all accessibility of our data visualizations.

4.4 Conclusion

Both of these examples only begin to scratch the surface of the potential discernibility of artist-supported palpable visualizations. This is primarily due to the fact that neither Weather Report nor our applications of Lift-Off were working with complex multi-variate datasets. But looking back at our collaboration with an artist during Bento Box back in Chapter 3, we saw that the limitation of expressiveness was not due to any shortage of inspiration on the artist's side, but rather in the simplicity of our own rendering techniques. As we move forward, we re-imagine how to develop visualization design software to leverage an artist's creativity in not only the spatial design of a data-driven experience, but also in the very data encodings themselves.

Chapter 5

A Theory and Implementation of Artifact-Based Rendering for Scientific Visualization

Harnessing Natural & Traditional Visual Media for Expressive 3D Vis

5.1 Introduction

[1] As we saw with our ensemble of volumetric simulation data in Chapter 3, designing a visualization that encodes multiple variables in a discernible way is a hard problem that can pique an artist's interest.

Through our exploration of creative design projects in Chapter 4, we learned that providing workflows that are accessible to the creative workflows of artists and designers can result in experiences that are more accessible to viewers than the traditional development of scientific visualizations.

Our key take-aways from this study are as follows:

- There are many potential contributions to be made by artists and designers to the study of visually encoding data

---

[1] This chapter is based on work published in IEEE Transactions on Visualization and Computer Graphics [Johnson et al. 2019b].

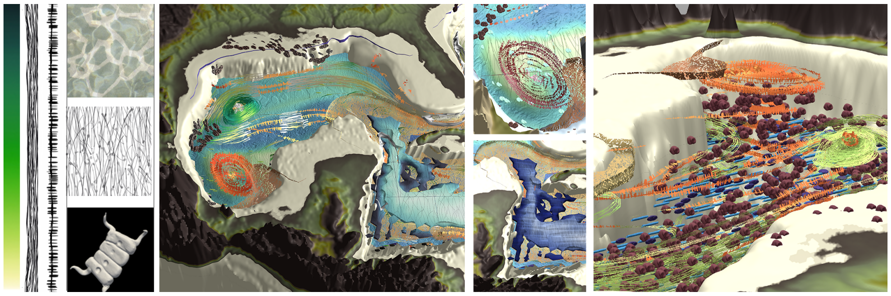


Figure 5.1: Using traditional physical artistic media as input to the digital visualization pipeline provides a richer visual vocabulary and opens the door for artists to participate in creating more expressive and engaging 3D scientific visualizations. This example helps scientists understand commercially viable macroalgae growth in the Gulf of Mexico by encoding temperature and salinity from remote sensing together with eddy direction and curvature and three nitrate concentrations from computational simulation.

- Existing visualization tools and workflows do not lend themselves to artist involvement
- Design is often most naturally begun using physical media and iterative processes
- The expressiveness of our visualizations to visually encode many attributes at once is limited by our rendering techniques and not on the creativity of a designer
- Visualizations based in the physical world are engaging to participants

We flesh out these points over this chapter and the next, describing our new approach to designing visualizations based on physical artifacts designed by artists for encoding multi-variate volumetric datasets called Artifact-Based Rendering (ABR) [Johnson et al. 2019b].[2] In this chapter, we walk through the theory and implementation of our ABR visualization design system. and in Chapter 6, we show how ABR can be applied to a

---

[2] Acknowledgements: This research was supported in part by the National Science Foundation (IIS-1704604 IIS-1704904). Brain microstructure applications were supported in part by the National Institutes of Health (P41 EB015894, P30 NS076408). MPAS-O simulations were conducted by Mathew E. Maltrud and Riley X. Brady as part of the Energy Exascale Earth System Model (E3SM) project, funded by the U.S. Department of Energy (DOE), Office of Science, Office of Biological and Environmental Research with analyses conducted by PJW, MEM, and RXB under ARPA-E Funding Opportunity No. DE-FOA-0001726, MARINER Award 17/CJ000/09/01, Pacific Northwest National Laboratory, prime recipient.

number of different datasets, and how we've thus far evaluated its usefulness to artists and scientists.

Finding inspiration in our physical world, combining disparate objects in new ways, understanding through hands-on making — for centuries these "low-tech", physical processes have helped us (scientists, artists, architects, doctors, engineers) to investigate, reinterpret, and ultimately make sense of our world. Visualization, particularly in immersive environments such as virtual or augmented reality (VR or AR), promises a similar, more physical, perhaps even innately human approach to making sense of today's complex data. Yet, current computer-based visualizations fall short of realizing many of the benefits of more traditional, time-tested, physical sense-making processes.

We highlight three specific challenge problems for the future of scientific visualization:

Challenge 1. Supporting traditional visual designers and design processes, such as those taught in art and design disciplines. Here,"supporting" means, in part, creating new visualization design tools that make it possible for non-programmers to rapidly design and critique many alternative data-to-visual mappings.

Challenge 2. Expanding the visual vocabulary used in visualizations in order to depict increasingly complex multivariate data. The consistent computer-generated aesthetic found throughout our conference proceedings is becoming well refined, but it is in sharp contrast to the much larger visual variety we find when walking the halls of a museum or even walking through the woods. Is it possible that our visual vocabulary is converging upon a local, rather than global, maximum? We ask, to what extent might a richer visual vocabulary increase overall expressiveness, with new visual encodings or new combinations of encodings conveying additional or more complex data?

Challenge 3. Bringing a more engaging, natural, and human-relatable handcrafted aesthetic to data visualization. We live in a time when there is a great disconnect between the scientific community and "lay people". Scientists struggle to tell their stories. In theory, visualization should be the most powerful tool scientists have to communicate with each other and the public, but the disconnect persists. Thus, we ask, how might even the methods we choose to depict scientific data reaffirm the natural, human connection to all aspects of science?

In formulating Challenge 3, we have closely followed recent research in data physicalization [Alexander et al. 2015; Djavaherpour et al. 2017; Khot et al. 2014; Taylor et al. 2015]. Benefits of data representations that are physical, real-world objects rather than purely digital representations include the potential to use multi-sensory perception and dual encodings to increase data legibility [Hogan and Hornecker 2016]. The potential benefits also include increased engagement and emotional connection with data [Jansen et al. 2015]. Imagine touching a physical, data-driven melting ice sculpture depicting in physical form how the terminus of Grewingk Glacier has receded over 150 years [Sengal 2015], and compare this to a typical online map-based visualization of the same data. Certainly a map will be better for some data analysis tasks, but the scientific community cannot dismiss the sculptural visualization. Cognitive science demonstrates that humans are innately compelled to touch and examine some physical objects [Klatzky and Peck 2012]. We engage and connect with natural, physical forms, especially when there is evidence of the human hand in them. Thus, a physical ice sculpture can be just as valuable to science and society as a finely crafted digital map and, perhaps, more impactful in the actions it inspires.

Our work seeks to extend physicalization techniques by introducing an inverse problem relative to what has been studied thus far, namely, using physicalization as an input to the visualization pipeline rather than just an output. We reason that the resulting, often handcrafted, aesthetic can have many benefits, including highlighting the human connection to the data. In a sense, this approach could mimic for 3D scientific data the hand-drawn 2D aesthetic Georgia Lupi has leveraged so effectively to advance her work on "data humanism" [2017].

Our work also closely follows research in visualization design tools, particularly those supporting artists' contribution to scientific visualization, which inspires Challenges 1 and 2. In the tradition of the "renaissance teams" introduced by Donna Cox [1988], our interdisciplinary team includes computer scientists, domain scientists, and a traditionally trained artist. Without new tools, it has been impossible to translate the multitude of stunning visual design ideas developed by the artist using traditional media (e.g., paper, ink, clay, wax) into 3D data-driven visualizations. Prior software systems and design processes have identified and addressed this same problem but incompletely. Scientific Sketching supports artists prototyping 3D visuliazations in VR, but not in a data-driven way [Keefe et al. 2008]. Vis-by-Sketching supports artists in data-driven prototyping, but only for 2D spatial datasets [Schroeder and Keefe 2016]. Our work is the next logical progression, extending to support rapid data-driven 3D visualization design and prototyping.

The major novel idea in this work is, therefore, to introduce a decidedly human, physical approach to crafting data visualizations by working with artists and the traditional media that they can so powerfully control (Figure 5.1). We demonstrate how all major visual elements of 3D multivariate scientific visualizations (color, line, texture, and form) can be derived from physical artifacts designed and crafted or even found in nature by these artists. Further, with a custom rendering engine, the artifacts can respond dynamically to data to produce complete, accurate, data-driven interactive visualizations. We call the framework of tools, algorithms, and processes that comprise this idea Artifact-Based Rendering (ABR).

The framework serves as a visualization design tool, but, to reiterate, the goal is not as simple as producing only efficient, effective solutions. A key anticipated benefit of ABR is opening the field to a more diverse group of visual designers who might contribute radically more effective solutions that have not yet been discovered. Since the foundation of creativity and design in traditional artistic fields is rapid exploration of alternatives (e.g., think sketching [Buxton 2010]); it follows that a key requirement for the framework is to support rapid exploration of many alternatives. ABR does this by leveraging the skills artists have with traditional media. While this may not be the most efficient solution for computer technologists to design visualizations, artists can create dozens of novel 3D glyph designs in clay in just one hour.

From a technical standpoint, several computing advances were required in order to present a first implementation of ABR. For example, we combine 2D and 3D scanning techniques, texture synthesis, and morphing in new ways for visualization while also supporting interactive 3D rendering. The main contributions of this chapter can be summarized as:

1. The concept and theory of Artifact-Based Rendering.
2. The design of four example front-end applets that prepare artifacts for visualization by constructing colormaps, cropping and calculating normal maps from image

artifacts, synthesizing new textures from examples, and optimizing 3D scanned meshes.

3. The specification for and implementation of a first ABR rendering engine with custom algorithms and interfaces to enable multiple new visual styles for depicting point, line, surface, and volume data.

To support it, we combine 2D and 3D scanning techniques, texture synthesis, and morphing in new ways for visualization. We also present an interactive rendering engine that combines traditional scientific visualization data processing, filtering, and transformation via the Visualization Toolkit (VTK) [Schroeder et al. 2004] along with interfaces and algorithms for depicting volumetric data fields in a variety of artifact-based styles. In Figure 5.1 and throughout the chapter we present results from applications to the actively researched data from domain science collaborators.

5.2 Related Work

5.2.1 Artistic Techniques and Theories in Visualization

The field of data visualization has often benefited from art and design theories, processes, and techniques. Examples include accentuating the legibility of 3D forms by rendering them in the style of pen and ink illustration [Winkenbach and Salesin 1994] or with shading based on artistic color theory [Gooch et al. 1998]. Inspired by traditional oil painting [Kirby et al. 2005], researchers have also developed algorithms to render data-driven "brushstrokes" for visualization that might translate to an improved ability to support multi-level understanding of data. Some results utilize painterly layering and composition but maintain a mostly geometric appearance [Kirby et al. 1999; Laidlaw et al. 1998]. In others, brushstrokes are clearly visible [Healey 2001; Healey and Enns 2002; Tateosian et al. 2007]. These look "painted" to most of us; however, artists critique the visualizations as regular and algorithmic, missing the richness and subtlety of a traditional painting.

Our work advances this research on three fronts. First, we extend to 3D the core concept of building a visualization up from data-driven, artistic, low-level visual elements. Second, we introduce a method for accomplishing this with real, physical artifacts rather than via algorithmically generated approximations, which are often limited in their ability to capture the original artistic intent. Finally, we include actual artists in the process of crafting these visualizations.

5.2.2 Artists and Designers in Visualization

The hand-drawn "Dear Data" series of 2D information visualizations [Lupi et al. 2016], demonstrate how artists themselves can develop their own visual languages to convey data in ways that are both accurate and inspire human connection to the underlying information. These results build upon a tradition of artists' contributing to visualization (e.g., [Cox 1988]) and advance goals of the National Academies of Sciences, Engineering, and Medicine, which call for expanded art-science collaborations [National Academies of Sciences et al. 2018].

New tools are required to support artist involvement [Kähler et al. 2002], and researchers have developed several. Drawing with the Flow [Schroeder et al. 2010] and later Visualization-by-Sketching [Schroeder and Keefe 2016] do this with custom pen-based user interfaces. Visualization-by-Sketching supports multivariate data layers and animated streaklets, but all in 2D. Volume Shop [Bruckner and Groller 2005] and WYSIWYG Volume Rendering [Guo et al. 2011] make designing transfer functions for volume renderings accessible to artists, but this does not quite address the multivariate design challenge in the sense that volume rendering is hard to extend beyond conveying 1 or 2 variables simultaneously. Scientific Sketching [Keefe et al. 2008, 2005] is something like a VR sketchbook for artists to design scientific visualizations. So, it is 3D and expressive, but it does not connect to any underlying data in order to turn the sketches into real, data-driven visualizations. Our work is the first in which artists may construct 3D data-driven visualizations using visual elements they craft themselves using the traditional media with which they are already experts.

5.2.3 Data Physicalization and Human Connection

Our approach is closely related to emerging research in "data physicalization" [Jansen et al. 2015]. Whereas data physicalization is the process of visualizing data via a physical output (3D printouts, sculptures, active touch tables, etc.), ABR is the inverse, using curated or handcrafted physical objects as inputs to generate digital data visualizations. In concept, this builds upon the work of artists, such as Mielbach [Samsel 2013], who use physical materials to provide context and connection for science. Both visual artists and psychologists have studied the geometric characteristics of shape (e.g., roundness, angularity, simplicity, complexity) and their impacts on human emotional responses [Lu et al. 2012]. There is evidence that data visualizations can be more effective and engaging when created by hand. Route maps can be more effective when presented in a hand-drawn style [Agrawala and Stolte 2001], and hand-draw iconography improves engagement and retention in data-driven storytelling [Lee et al. 2013; Rogers et al. 2017].

Glyphs (see surveys [Fuchs et al. 2017; Ropinski et al. 2011; Ward 2002]) are one area where we believe ABR can make a powerful contribution to visualization. Prior glyph designs are characteristically geometric in their visual aesthetic. This is true for glyphs formed by superposition of 3D primitives (cones, spheres, cubes) [Feng et al. 2009; IV et al. 2002; Legg et al. 2017; Lombeyda 2016] or parametric surfaces that are elegantly defined to vary in response to multiple data axes [Kindlmann 2004]. In contrast, our collaborative project grew out of discussions of how an artist would approach designing similar glyphs. The discussion quickly turned to a demonstration, when a box of 250 small handcrafted clay glyph "sculpturettes" arrived by mail from the artist. With a bit of clay in hand, a sculptor can create 40 to 50 alternative designs for 3D glyphs that could be used for multivariate flow visualization within an hour.

5.2.4 Colormaps and Textures for Visualization

Seminal research on designing and testing perceptually accurate colormaps for general use in visualizations [Moreland 2009; Rheingans 2000; Rogowitz and Kalvin 2001; Ware 2012; Zhou and Hansen 2016] establishes several rules of thumb, such as relying primarily on luminance and saturation for depicting magnitude data. Tools are also available for evaluating and modifying maps to adhere to commonly accepted guidelines [Bujack et al. 2018; Kovesi 2014; Zhou and Hansen 2016], which closely parallel the fundamentals of color theory as studied by artists [Albers 2009; Itten and Birren 1970]. Our work shares a similar motivation with systems that leverage artistic color theory, example works of art, or artists themselves to design effective colormaps. Our Color Loom applet extracts the color palette automatically from source images, which can be works of art, similar to

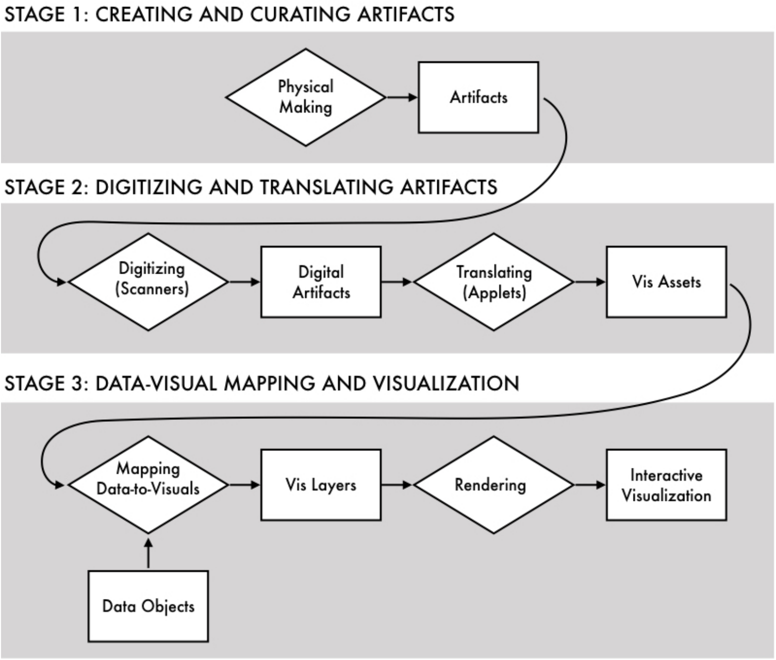

STAGE 1: CREATING AND CURATING ARTIFACTS
Physical Making
Artifacts
STAGE 2: DIGITIZING AND TRANSLATING ARTIFACTS
Digitizing (Scanners)
Digital Artifacts
Translating (Applets)
Vis Assets
STAGE 3: DATA-VISUAL MAPPING AND VISUALIZATION
Mapping Data-to-Visuals
Vis Layers
Rendering
Interactive Visualization
Data Objects

Figure 5.2: The ABR pipeline contains three main stages.

manual approach introduced by Vote et al. [2003]. Because they are quick to create, and artists can tune the results based on the data, the resulting colormaps are often useful for revealing more information in specific datasets [Patchett et al. 2016; Samsel et al. 2015] or even better engaging users, as in recent studies of affective use of color [Samsel et al. 2018].

Although not as common as colormapping, varying texture in response to data is a technique that has also been used previously [Healey and Enns 1999; Laidlaw et al. 1998; Ware and Knight 1995], including to encode uncertainty [Botchen et al. 2005]. Closely related to our work is that of Interrante et al. who encoded data using natural 2D textures of fibers and weavings of different densities [2000]. Later Gorla et al. extended this to synthesize texture from an example that follows a vector field on a 3D surface [2003]. We identify data-driven synthesis as important step for future work.

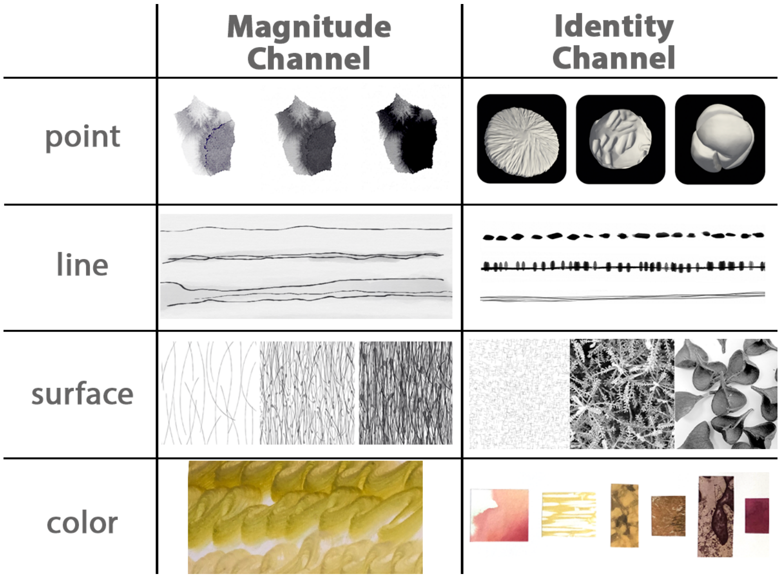


Figure 5.3: Artists will recognize the formal properties of (point, line, form, texture, and color) in these visual examples. Visualization scientists will recognize magnitude channels to encode ordered data and identify channels to encode categorical data. The volume category focuses on color schemes for volume rendering algorithms.

5.3 Artifact-Based Rendering for Visualization

ABR is a framework of tools, algorithms, and processes that makes it possible to produce real, data-driven 3D scientific visualizations with a visual language derived entirely from colors, lines, textures, and forms created using traditional physical media or found in nature. This section presents the ABR theory, processes, and technical system details that have been developed through a two-year, iterative process.

Figure 5.2 diagrams the full pipeline for ABR visualization, divided into three stages: (1) Physical design work to craft artifacts; (2) Digitizing and translating artifacts for data-driven visualization; (3) Creating data-to-visual mappings to implement multivariate interactive visualizations. Given the current library of pre-loaded artifacts, it is possible to begin a new project at any stage and then return to earlier stages as needed to create or adjust artifacts.

5.3.1 Stage 1: Creating and Curating Artifacts

Stage 1 of ABR is concerned with making or curating **artifacts**, physical representations of color, line, texture, or form that are the elemental visual building blocks of the final visualizations. Artists are used to thinking in these terms. Artists build, analyze, and deconstruct visual scenes using design elements known as *formal properties*: point, line, shape, form, texture and color [Evans and Thomas 2008; Lauer and Pentar 2012].

Recognizing the similarity in the way artists define formal properties and the way scientific visualization practitioners characterize the topology of underlying data variables (point, line, surface, volume), we organize artifacts as diagrammed in Figure 5.3. Notice that artifacts are grouped not just by data topology, but also by use, following Munzner's classification [2014] that distinguishes between visual marks for effectively encoding "magnitude" relationships (e.g., scalar temperature data) and marks for encoding "identity" relationships (e.g., phytoplankton vs zooplankton).

Artifacts may be sculpted with 3D artistic tools; we have experimented with clay sculpturettes, imprints, and shaved wax. 2D artifacts are also useful, and we have experimented with drawing, painting, texture rubbings, prints, and photography. Finally, artifacts can be acquired from an endless number of found objects, and these can be arranged in patterns to produce even more vis assets. We have worked with leaves, rice, lentils, rice noodles, seed pods, gravel, photographs of friends' paintings, and found photographs. Color artifacts can come from many sources, including painting and collage.

5.3.2 Stage 2: Digitizing, and Translating Artifacts

Stage 2 of ABR begins with capturing the material appearance and/or form of the physical artifacts produced in Stage 1 as **digitized artifacts**. Here, the specific capture technique depends completely on the type of artifact and sometimes also on the intended eventual use. Next, digitized artifacts are translated into **vis assets**, for example, color maps defined in a standard color space, computer graphics-ready textures, glyph meshes that are correctly oriented, down-sampled if necessary, and have normal maps applied for fast rendering.

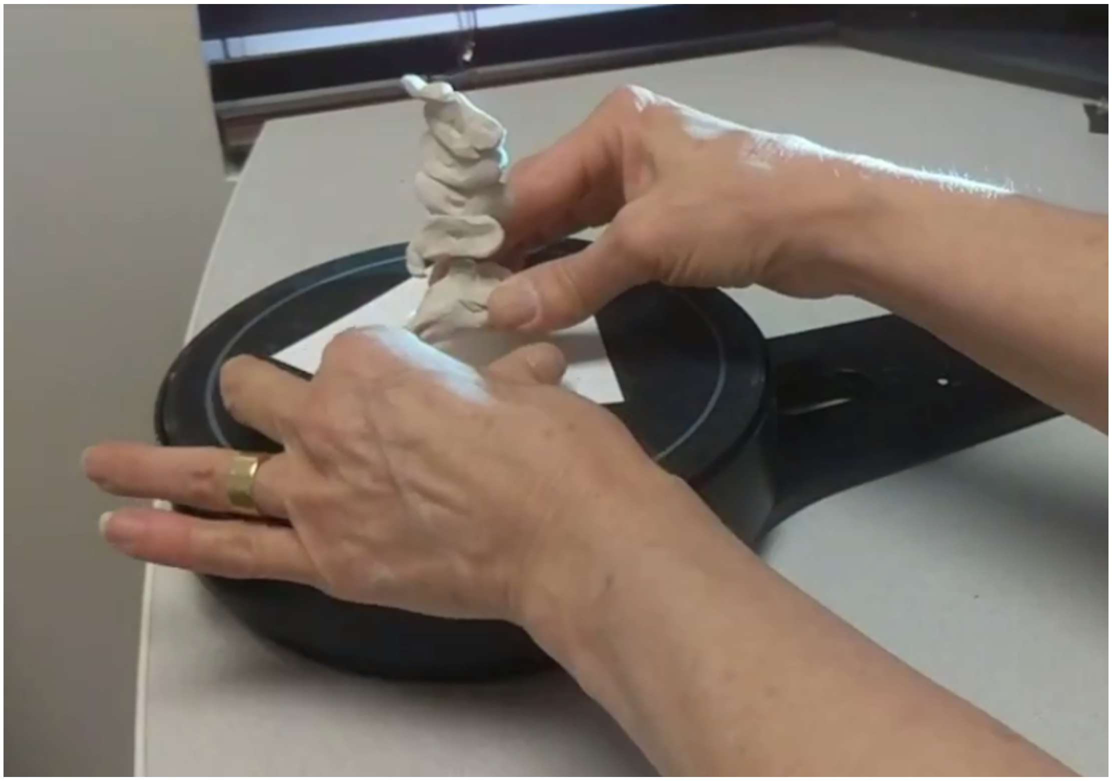


Figure 5.4: The EinScan-SE structured light 3D scanner makes 3D scanning of physical artifacts reliable and reproducible.

Digitizing Material Appearance and Form

A variety of capture techniques can be used, and the current implementation demonstrates several options for both 2D and 3D artifacts. Photography (e.g., digital photographs of seed pods and bark found in nature), scanning (e.g., scans of hand-painted ink wash lines or painted color maps), or digital tools with physical inputs (e.g., drawing boards) are used to capture handcrafted 2D material appearances. In the future, we are keen to also include lighting dependent material appearances using methods, such as linear light source reflectometry [Gardner et al. 2003].

Photogrammetry and structured light 3D scanners are used to capture 3D material appearance and form. Most of the digital artifacts pictured here were captured using an automated EinScan-SE structured light 3D scanner. Each scan produces a high-resolution mesh of around 100,000 vertices with corresponding photographic texture data. In the future, low-cost smartphone based scanning might also be used (e.g., [Muratov et al. 2016]). For our work, however, the EinScan-SE hardware made 3D scanning a repeatable process our artist could perform herself (Figure 5.4), and produces a high-fidelity `.obj` file with a poly-count on the order of 100,000 triangles.

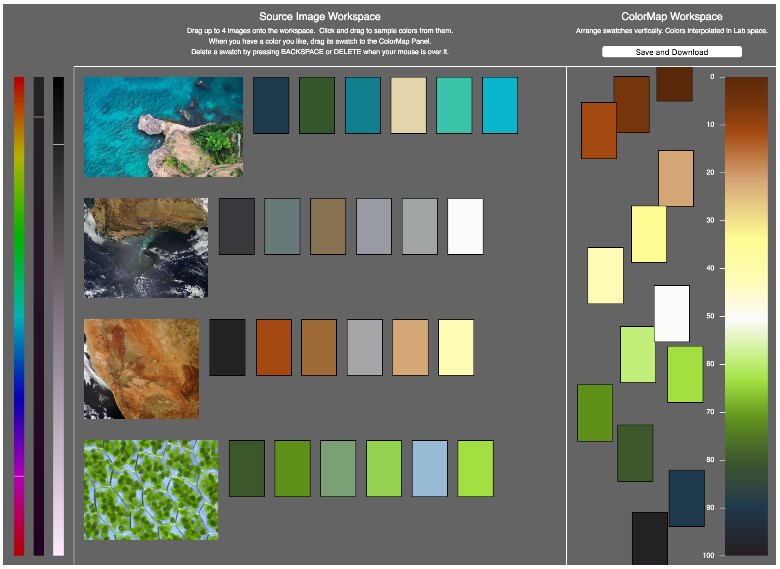
While the `.obj` files are of a high quality, they are scanned without any particular orientation, and their high vertex count greatly limits the number of glyphs that can be rendered at interactive framerates on a consumer computer. Thus, we implemented a set of scripts to align and reduce the glyphs to prepare them for rendering, which you can read more about in Section 5.3.2.

An open online digital library and underlying database system stores the raw digitized artifacts and the vis assets that are generated from them in the next step. Metadata classifying the artifacts based on material type and possible use (e.g., to encode line, direction, points) are included to enable online searching and filtering.

Translating to Vis Assets

Digitized artifacts are translated to vis assets using four custom interactive applets. These are needed for two reasons. First, processing is often required before raw scans can be used in the rendering pipeline. Second, we wish to be able to reinterpret each raw digitized artifact in multiple ways as a vis asset (e.g., a scanned 3D mesh might be used both to define a 3D glyph shape *and* a normal map for a bumpy texture to apply to an isosurface). User interactions with the four applets are demonstrated in the accompanying video.

Applet 1: Color Loom. The Color Loom applet (Figure 5.5) helps artists to weave colors from digitized photographs, paintings, and other artifacts together into coherent colormaps. First, the artist drags-and-drops one or more source images onto the left panel of the window. A modified median cut quantization algorithm [Bloomberg 2008; Heckbert 1982] then identifies a suggested palette of six prominent colors found in the image, which are displayed as editable color swatches to the right of the source image. Artists can manually pull additional colors from the source images and create more swatches by hovering over specific pixels. The color of any swatch may be tweaked using hue, saturation, and brightness sliders. To build a colormap, swatches are dragged to the right panel, where their vertical positions define control points for a continuous



Drag up to 4 images onto the workspace. Click and drag to sample colors from them.
When you have a color you like, drag its swatch to the ColorMap Panel.
Delete a swatch by pressing BACKSPACE or DELETE when your mouse is over it.

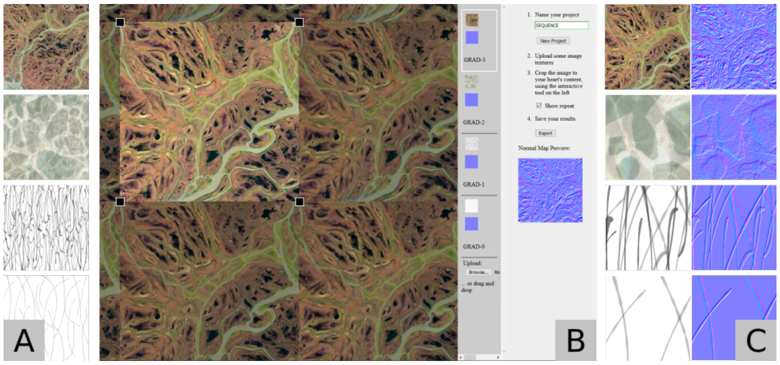
Arrange swatches vertically. Colors interpolated in Lab space.

Figure 5.5: The Color Loom applet. Artists drag and drop source images into the left panel and pull swatches of color from these, which are then copied and arranged in the right panel to create a color map.

colormap with interpolation performed in CIE Lab space. Results are saved in Paraview .XML and PNG image formats.

Applet 2: Texture Shaper. The Texture Shaper applet (Figure 5.6) supports cropping, previewing repeating texture patterns, and saving results in a standard format expected by the ABR rendering engine. It also supports more advanced features that are useful for ABR, including converting source imagery into normal maps that can be used in per-pixel lighting calculations for 3D rendering and building ordered texture sets for encoding data, like a binned gradient. Results are exported from the applet as (sets) of compressed PNG image(s).

Applet 3: Infinite Line. The Infinite Line applet (Figure 5.7) uses texture synthesis to transform an example image of a vertical linear mark into a longer, seamlessly repeating texture that can be mapped onto ribbons and other 3D forms. The algorithm follows the "video textures" algorithm presented by Schödl et al. [2000], which has been used to synthesize drawn and painted strokes for computer graphics non-photorealistic rendering [Kalnins et al. 2002] and is a natural fit since only 1D texture synthesis is required. A similarity measure is computed, comparing each row of the texture to every



1. Name your project
SEQUENCE
2. Upload some image textures
3. Crop the image to your heart's content, using the interactive tool on the left
☑ Show repeat
4. Save your results
Normal Map Preview:
... or drag and drop

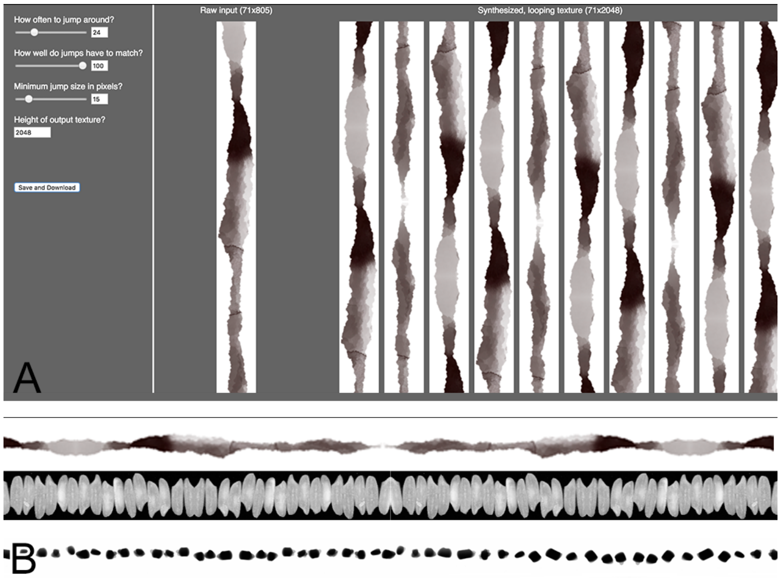
Figure 5.6: The Texture Shaper applet. A: Original source images, B: Selecting a cropping box; C: Output images and normal maps.

other row. Then, a new texture is synthesized, starting from a random starting row and proceeding through the texture by either moving to the next row in the original source or, with some probability, jumping to a new similar row. A heuristic is used to make the final image loop seamlessly; after synthesizing an image five times larger than the desired output height (typically 2048 pixels), the algorithm searches through the result to find the subsection of the desired height where the starting and ending rows are most similar.

Since outputs of texture synthesis algorithms like this one are highly dependent upon the algorithm's parameters (e.g., probability of "jumping" to a new row, the minimum allowable quality for "jumps", the minimum size of a "jump"), the applet makes it possible for artists to adjust these parameters and view the results in real time, preserving the visual characteristics that encode identity while avoiding the distracting regular pattern that is visible with a regular, repeating tiled texture.

Applet 4: Glyph Aligner. The Glyph Aligner applet (Figure 5.8) works with 3D scanned artifacts, which require user input and data processing before being used as vis assets. On the left, mouse-based camera and object trackball controls are used to reorient the glyph to associate "forward" and "up" directions with the 3D scan. On the right, a preview updates in real-time to show the result of applying the glyph to visualize an example vector field.

Before they can be reoriented via the browser applet, the 3D-scanned `.obj` files are first uploaded to a P5-based webpage, and an initial decimation is performed through



How often to jump around? 24

How well do jumps have to match? 100

Minimum jump size in pixels? 15

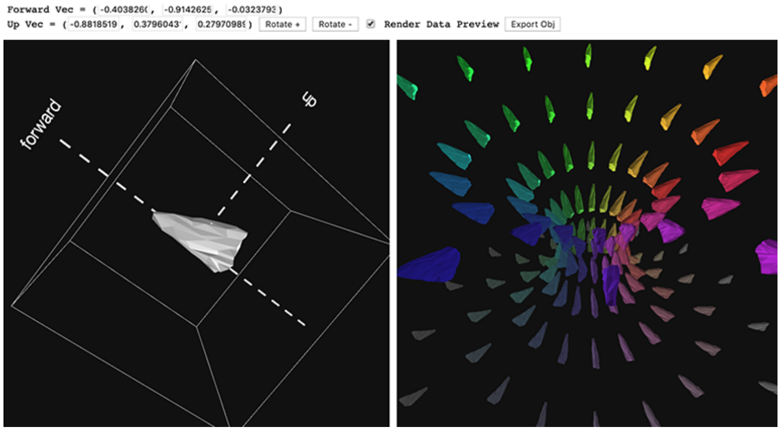
Height of output texture? 2048

Figure 5.7: The Infinite Line applet. A: The user interface with parameter controls and texture synthesis preview. B: Examples of textures synthesized from inkwash, rice grains, and ink dots.

a server-side blender script to reduce the poly count so that the object can be rendered interactively in the web-page. Then the user can then rotate the model to specify a forward and up direction, and the transformation matrix is then applied to the original high-poly model, transforming the vertices to the new alignment.

After interactively aligning the model, the resulting mesh is passed to an automated Blender3D Python script. This script decimates the 3D scanned mesh, which may include 100,000+ vertices, to varying degrees to support level-of-detail (LOD) rendering in VR, while preserving detail with normal mapping. A UV mapping for each mesh is creating using Blender's "Smart UV Project" algorithm, and normals are stored based on the original geometry. Then, for each LOD mesh the differences between these original normals and the normals of the decimated mesh are baked into a LOD-specific normal map. The output can reduce the vertex count by three orders of magnitude while preserving much of the surface appearance (Figure 5.9).

The resulting low-poly meshes are stored in a database with their corresponding normal maps, which can be loaded into Unity3D for instanced-rendering with standard Unity3D surface shaders.



Forward Vec = (-0.403826 , -0.9142625 , -0.0323793)

Up Vec = (-0.8818519 , 0.3796043 , 0.2797098)

☑ Render Data Preview

Figure 5.8: The Glyph Aligner applet. Artists use trackball controls in the left panel to align a 3D scanned glyph. The right panel provides a glyph field preview using synthetic data.

5.3.3 Stage 3: Data-Visual Mapping and Visualization

Stage 3 of ABR involves implementing data-driven interactive visualizations using the vis assets. To provide structure to these multivariate visualizations, we say that each visualization is composed of multiple **vis layers**. These are analogous to the 2D layers used by artists in image editing programs, but vis layers are not 2D; they are true 3D volumetric constructs. Each vis layer has hooks for connecting vis assets to **data objects**. Thus, after creating a new oriented glyph layer, a designer could attach vis assets to define the 3D glyph mesh and color map to use for the vis layer and also attach data objects (e.g., density sampled phytoplankton concentration, velocity magnitude) to drive color changes. The layers are combined into a final **interactive visualization**, which may be rendered fast enough to optionally display in head-tracked, stereoscopic VR.

Defining a Specification for an ABR Rendering Engine

We begin by more completely defining the problem an ABR rendering engine must solve. An ABR rendering engine must produce 3D computer graphics imagery of various data topologies (points, lines, surfaces, and volumes) using, as directly as possible, the visual styles defined by real-world physical artifacts. This is a hard problem because it

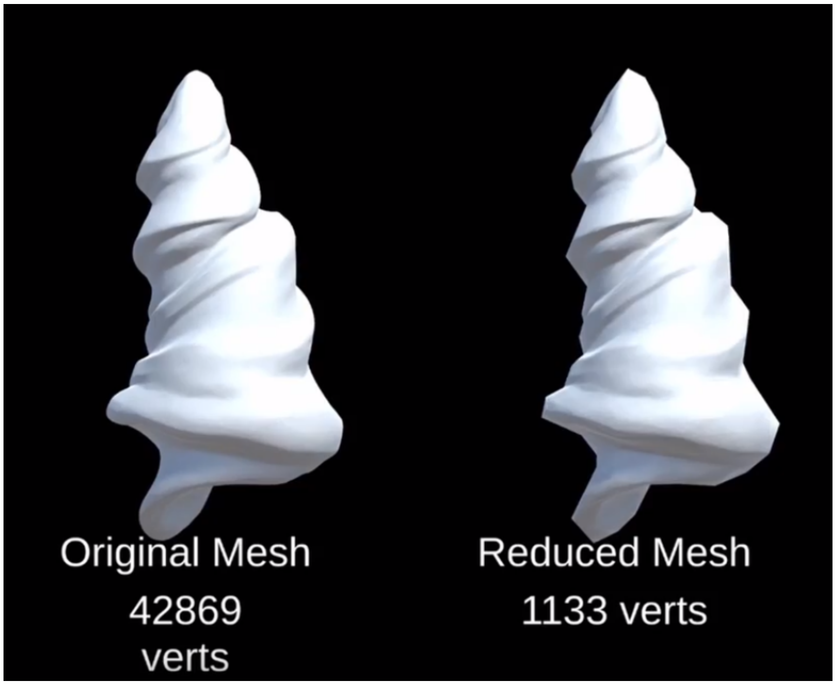
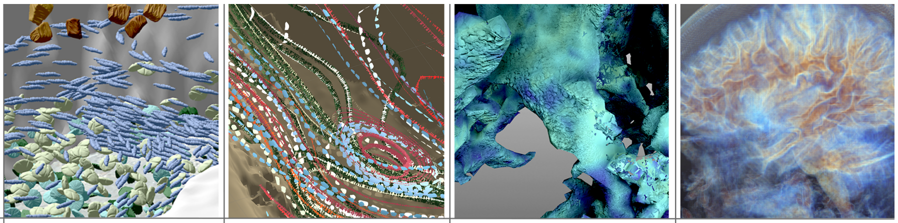


Figure 5.9: The EinScan-SE structred light 3D scanner makes 3D scanning of physical artifacts reliable and reproducable.

requires a balance between staying true to the original visual properties of the artifacts and manipulating these based on underlying data. Also, artifacts can be interpreted in so many different ways. Given a series of evocative, organic textures captured from leaves, rocks, or seeds, how precisely should they be used to encode data categories or magnitudes? The answer will likely change based on the data topology (points, lines, surfaces) and on the goals of the visualization. Thus, we reason that, like all good design tools, a good ABR rendering engine should provide options (multiple complementary rendering techniques) for how to interpret artifacts and attach them to data. We think of the possible techniques as existing along a spectrum.

Use Artifacts Directly as Assets
Infer Style from Artifacts & Synthesize Assets

On the left are ABR rendering techniques that can be implemented with little or no change to traditional visualization rendering. Color is one example. Existing systems



data object	3D point set (list of points) direction variable (vector) color variable (scalar) width variable (scalar)	3D line set (list of poly lines) color variable (scalar) texture vline width alpha mask (scalar)	3D surface mesh color variable (scalar) texture variable (scalar) alpha mask (scalar)	3D image stack color variable (scalar) opacity variable (scalar) alpha mask (scalar)
vis assets	3D glyph mesh colormap texture stamp alpha mask image normal map image	colormap texture stamp alpha mask normal map image	colormap texture stamp images (list) alpha mask images (list) normal map image (list)	colormap opacity map
additional parameters	glyph size max - min scale random rotation max glyph	shape (ribbon) line width rotational offset texture-blending variable (scalar)	texture blending factors texture scalar factors	opacity multiplier

Figure 5.10: Example renderings and parameters for vis layers in the ABR rendering engine.

typically support rendering a constant color per data object for identity encodings, or color mapping for magnitude encodings. If the colors come from real-world source images, this can be considered an ABR rendering technique, albeit at the far left of the spectrum.

Beyond color, most ABR rendering techniques are non-trivial to implement in available visualization rendering engines. For example, existing tools often limit 3D glyphs to preset geometric primitives. Textured lines or surfaces are not difficult to implement from a computer graphics standpoint, but scientific visualization engines do not typically expose an ability to set the texture of lines and surfaces in order to support identity encodings, and we know of no existing 3D scientific visualization design tools that support varying these textures to support magnitude encodings. We demonstrate each of these "middle of the spectrum" techniques.

On the right of the spectrum, we predict specific ABR rendering techniques may themselves be the subject of future computer graphics research. For example, given a set of clay glyphs used to encode twist along a line, computer vision and machine-learning algorithms could be used to infer to visual style implied by the artists' examples and synthesize new parametrically controllable forms. The Infinite Line applet described earlier is an important step toward this goal of automated synthesis of vis assets from artifact examples.

ABR rendering techniques must support both identity channel and magnitude channel encodings (see Figure 5.3). Identity encodings are generally easier. Magnitude encodings are more innovative, and there are at least three possible approaches, ordered moving from left to right along the spectrum: (1) Mapping an ordered set of artifacts piecewise to data, like a binned color map. (2) Using artifacts as control points, implementing some form of morphing between them, and mapping the results to continuous data. (3) Parameterizing some characteristic of an artifact and synthesizing new instances with data controlling this parameter. This could be as simple as controlling the width of a glyph artifact, or as complex as synthesizing new texture patches with higher density distributions of a pattern in response to underlying data.

Mapping Data to Visuals in Layers

Building upon metaphors that work for artists in other visual design tools (e.g., image editors), we think of each new visual addition to the scene as a layer, many of which can be seen in Figure 5.16. Every vis layer takes as input one data object, some number of scalar or vector variables of that data object, and some number of vis assets, all of which may be selected at design-time by the user (Figure 5.10). The extensible rendering engine currently supports four layers.

Glyph Vis Layer. The glyph vis layer renders instances of a glyph (either a 3D mesh or an image displayed on data-aligned quads) located at coordinates specified by a point set data object. The glyphs can be axis-aligned, aligned to a selected direction vector variable, or assigned random orientations. Color can be assigned either as a constant across all glyphs or data-driven. A constant glyph size can be selected (defined as a percentage of the largest extent of the data object), and an axial radius can be specified either as a constant, or according to a selected width scalar variable. Normal maps are automatically applied if available. For image-based glyphs, alpha masks may be supplied to mask away the negative space.

We found that an important use for custom-sculpted glyphs is to represent scalar field data by distributing glyphs on a surface or throughout a volume. Thus, regular, random, and density-based samplings are supported. For density-based sampling, we implemented a Metropolis-Hastings algorithm [Chib and Greenberg 1995], which is a Markov Chain Monte Carlo method.

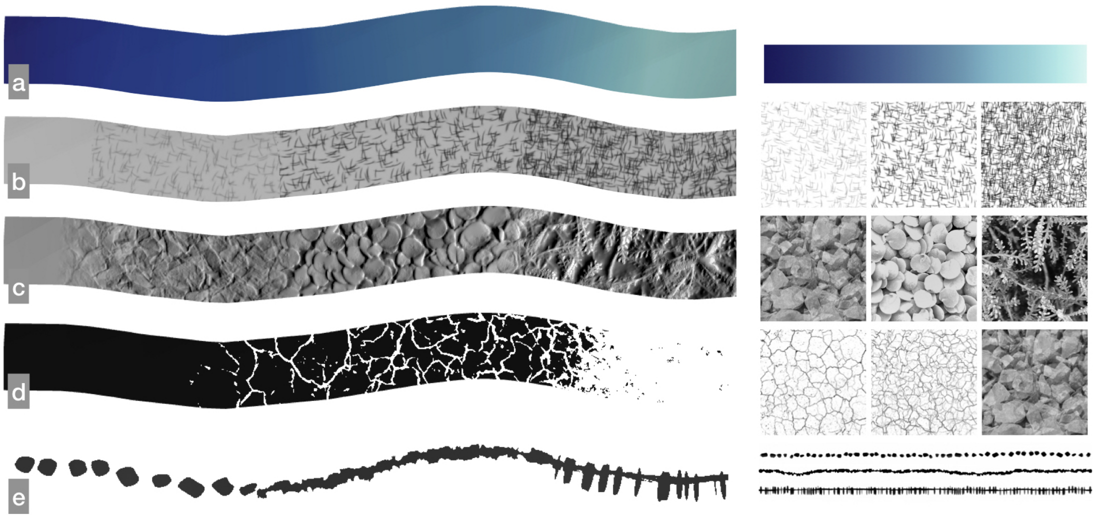


Figure 5.11: Options for applying vis assets to lines or surfaces include (a) color mapping, (b) data-driven texturing, (c) data-driven texturing with bump mapping, (d) data-driven texturing with blending and masking, (e) data-driven texturing with masking to create an organic line profile.

Line Vis Layer. The line vis layer renders ribbons or tubes along paths defined by a line set data object. If normal vectors for each point are provided, the ribbon or tube can be oriented according to the line normal and some rotational offset, and each mesh vertex is assigned texture coordinates with u equaling the arc-length from the line origin to the vertex, and v running from 0 to 1 either across the ribbon, or along the circumference of each tube ring. Color is assigned either as a constant along the entire geometry or based on a data-driven colormap. Similarly, a single tiled stamp, alpha-mask, or normal map can be applied across the entire geometry.

Optionally, an ordered texture set can be used for data-driven texturing. In this mode, textures are applied using a binned data mapping; the data range is divided into N evenly-sized bins, where N is the number of textures in the corresponding texture set. Blending can be applied in the fragment shader to hide texture seams, with a user-defined blend distance. Figure 5.11b-e shows specific examples of how textures are binned and blended.

Our implementation supports multiple sampling strategies and artists can switch between samplings that have equal steps in arc length or integration time, making it possible for the visual style to include evenly placed glyphs that are distorted (stretched) to encode speed or to include undistorted glyphs that are spaced along the line to encode speed.

Surface Vis Layer. The surface vis layer renders a triangle mesh defined by a polygonal mesh data object. Every fragment of the surface can be colored and textured based on either constant vis assets, or vis assets that are blended by the provided data variables. Since the surfaces are often complex and have no inherent UV texture parameterization, an automatic texture mapping approach is needed. We implemented a tri-planar projection technique, where the texture is projected along the axes and the three projections are blended according to the normal of the surface at any given location. This blending is further influenced by a projection blending factor that controls how crisp the seams are between the three projections. If the texture has a clear structure, such as grains of rice, a low blending factor may be preferred, whereas if the texture is more continuously varying, such as watercolor strokes, a higher blend factor will help hide the seams. All of line-style effects shown in Figure 5.11a-d can also be applied to surface layers to do data-driven texturing on arbitrary 3D surfaces.

Volume Vis Layer. Artists often refer to density fields of glyphs created with Glyph Layers as a “volumetric effect”. However, the engine also supports visualizing volumetric scalar fields using traditional volume rendering. The volume vis layer volume-renders a 3D grid of voxel data using artifact-based colormaps as a transfer function.

Implementation of Layers

Although it may be possible to implement ABR on top of a variety of other existing rendering pipelines using the concepts and framework described here, before this work, there was no existing system that could support the rendering required without a significant system-building effort. Our implementation is built upon a combination of VTK, for access to advanced data processing routines, and Unity, for better support for rendering in a variety of immersive displays and future support for 3D and tangible user interface techniques. All of the software, artifacts, and assets are being made openly available to the public.

A vis layer is a mapping between vis asset(s) and a data object that can be rendered to the 3D computer graphics scene. A vis layer may be as simple as drawing a one-pixel wide line along the shape of a line data object with a given color, or as complex as applying several vis assets to vary multiple data-driven visual properties at every point along the line. Every vis layer takes as input one VTK data object, some number of scalar or vector variables of that data object, and some number of vis assets, all of which may be selected at design-time by the user. Whenever a parameter is changed, the vis layer triggers its pre-processing routine to convert the data variables and vis assets into renderable Unity3D meshes and textures. If these Unity3D assets can be represented as game objects in Unity3D's built-in scene-graph, rendering is handled automatically. Alternatively, custom draw calls can be made each frame for more advanced rendering techniques, including writing layer-specific custom shaders.

Vis layers are cheap to develop. At minimum, a new class is defined inheriting from a `VisLayer` base class, providing an implementation for an asynchronous abstract method that is used to generate to persistent rendering objects and a rendering strategy. Vis layer parameters are declared as C# properties with a `[LayerParameter(string paramaterName)]` attribute, which are monitored via reflection to generate layer-specific user interface panels and trigger updates of layer rendering objects according to changes. Vis layer parameters can be any type, but will most often be primitives (floats, ints, bools), vis assets (Textures, Meshes), or data objects.

During the course of the research, we developed many separate `VisLayer` classes, each investigating a different ABR technique (e.g., `SimpleLineVisLayer`, `BinnedTextureRibbonVisLayer`, `BlendedProfileRibbonVisLayer`). Now, after honing the techniques, we have been able to refactor to group the algorithms into the following four main `VisLayer` classes, examples of which are pictured in Figure 5.10.

Glyph Layer. The glyph vis layer renders instances of a glyph (either a 3D mesh or an image displayed on data-aligned quads) located at coordinates specified by a VTK point set data object (Figure 7.17). The glyphs can be axis-aligned, aligned to a selected direction vector variable, or assigned random orientations. Color can be assigned either as a constant across all glyphs or data-driven. A constant glyph size can be selected (defined as a percentage of the largest extent of the data object), and an axial radius can be specified either as a constant, or according to a selected width scalar variable. Normal maps are automatically applied if available. Profile images are treated as luminance masks, with a corresponding brightness cutoff, which cuts away

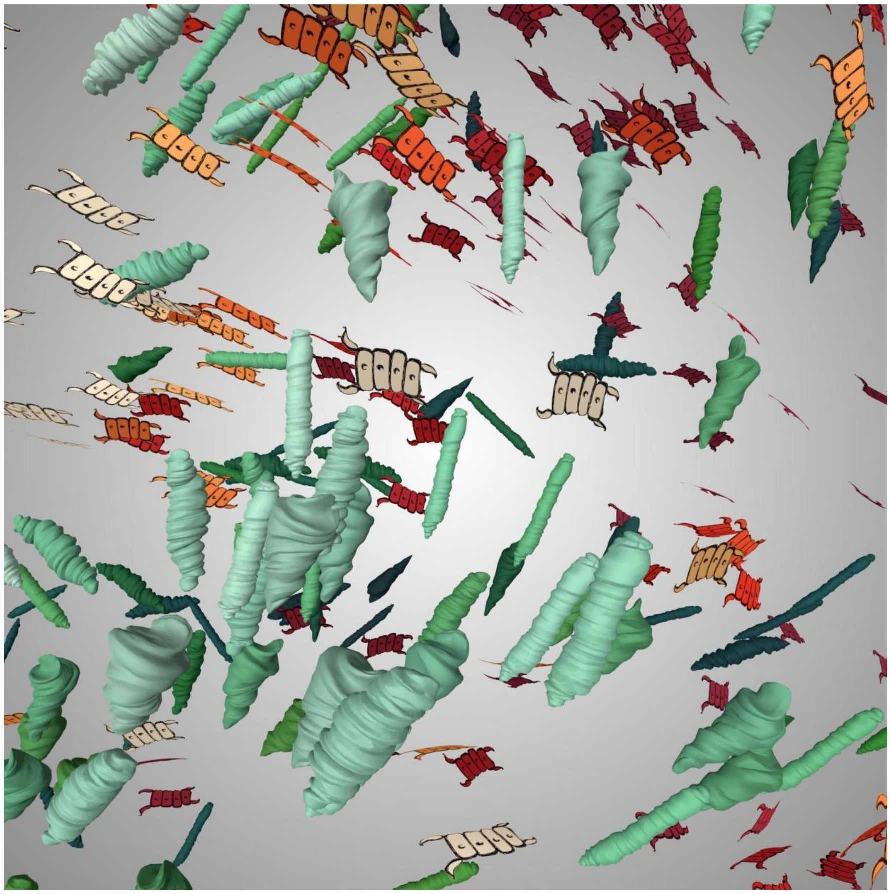


Figure 5.12: Two examples of surface layers using color and aspect ratio to encode multiple variables in both 3D glyph and 2D glyph styles.

the negative space of an image glyph corresponding to either the brightest or darkest regions of the profile image.

During the render object creation routine, per-glyph color and transformation attributes are calculated and stored in GPU arrays. Then on each `Update()` call, the glyphs are drawn using a batched instanced rendered call, with a shader that renders each mesh with the specified color, alpha, and normal maps, and spatial transforms.

Line Layer. The line vis layer renders ribbons or tubes along paths defined by a VTK line set data object (Figure 5.13). If normal vectors for each point are provided, the ribbon or tube can be oriented according to the line normal and some rotational offset, and each mesh vertex is assigned texture coordinates with u equaling the arc-length

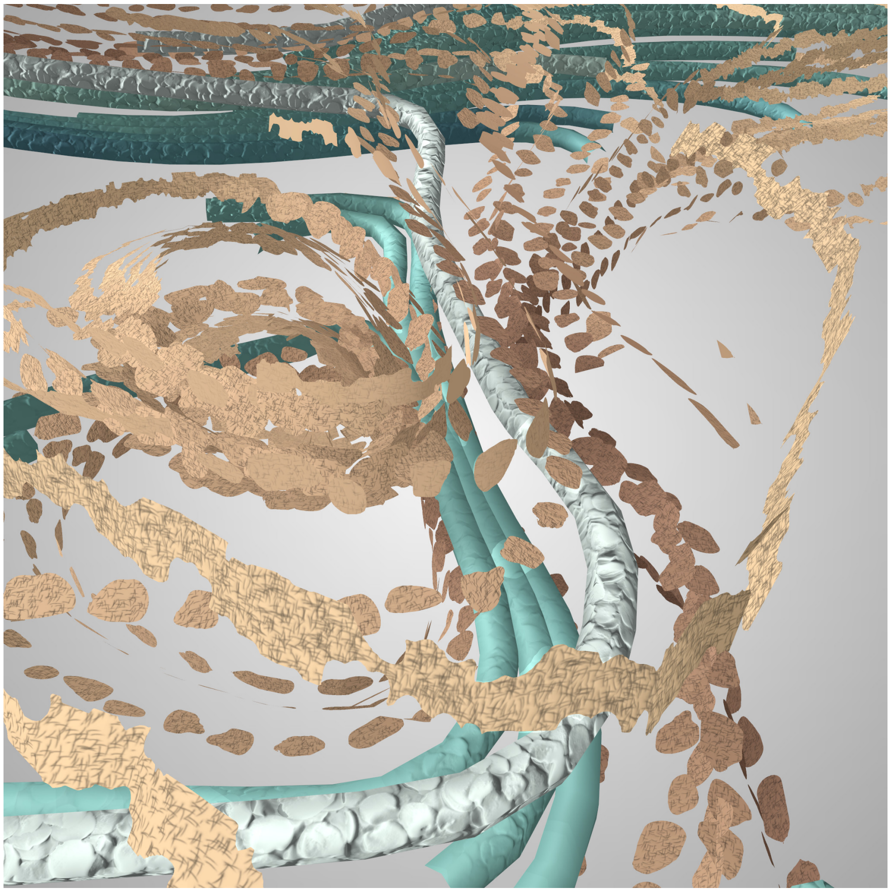
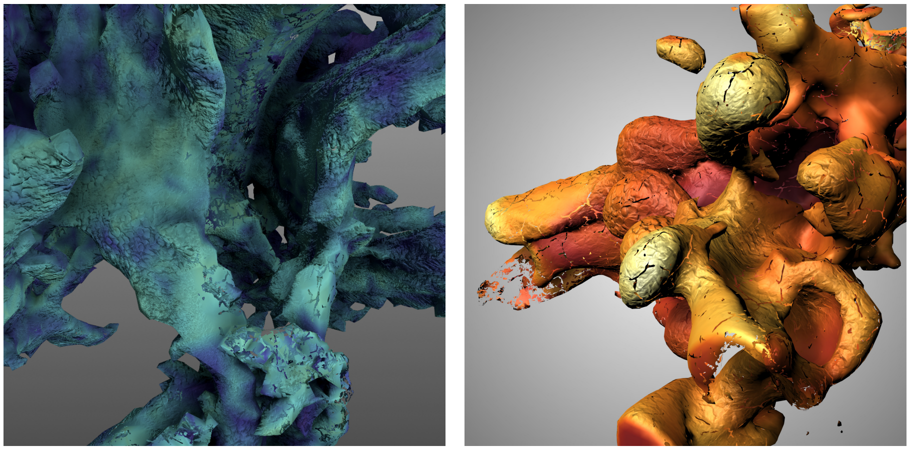


Figure 5.13: Two examples of line layers using color and texture to encode multiple variables in both ribbon and tube line styles.

from the line origin to the vertex, and v running from 0 to 1 either across the ribbon, or along the circumference of each tube ring.

Color is assigned either as a constant along the entire geometry or based on a data-driven colormap. Similarly, a single tiled stamp, alpha-mask, or normal map can be applied across the entire geometry, or an ordered texture set can be used for data-driven texturing.

If a texture set is used, textures are applied using a binned data mapping; the data range is divided into N evenly-sized bins, where N is the number of textures in the corresponding texture set. Blending can be applied in the fragment shader to hide texture seams, with a user-defined blend distance.



(a) A surface from the brain data (b) A surface from the astrophysics data

Figure 5.14: Two examples of surface layers using color, texture, and alpha masking to encode multiple variables.

Figure 5.11b-e shows specific examples of how textures are binned and blended, using color (a) or texture (b-d). In (b) the bin-blend value is zero, and the seams between the textures are visible; (c-e) use non-zero values. A normal texture is used in (c) to better capture the material appearance of the original artifacts. Alpha masks are used in both (d) and (e) to cut-out profile shapes in the line.

A custom shader is used to implement this ABR rendering technique. During the render object creation routine, new Unity3D meshes are constructed, storing the datarange-normalized variable values for the colormap, stamp, and normal map texture, and alpha mask texture into the r,g,b channels of the vertex color attributes respectively. The fragment step of the shader performs the bin-blending and application of all the textures, multiplying the selected color with the luminance of the stamp texture, and discarding the fragment if the luminance of the alpha mask falls below the specified cut-off.

Surface Layer. The surface vis layer renders a triangle mesh defined by a VTK polygonal mesh data object. Every fragment of the surface can be colored and textured based on either constant vis assets, or vis assets that are blended by the provided data variables (Figure 5.14). Since the surfaces are often complex and have no inherent UV texture parameterization, an automatic texture mapping approach is needed. We implemented a tri-planar projection technique, where the texture is projected along the axes and the three projections are blended according to the normal of the surface at any given location. This blending is further influenced by a projection blending factor that controls how crisp the seams are between the three projections. When this factor is 0, no blending occurs, and the three projections have crisp separation. When the blending factor is 1, every fragment contains some weighted combination of all three projected images. If the texture has a clear structure, such as grains of rice, a low blending factor may be preferred, whereas if the texture is more continuously varying, such as watercolor strokes, a higher blend factor will help hide the seams. All of the effects shown in Figure 5.11a-d can be applied to surface layers to do data-driven texturing on arbitrary 3D surfaces.

Volume Layer. The volume vis layer generates a traditional volume-rendered image of a 3D grid of voxel data using artifact-based colormaps. Our Unity3D CG shader implementation performs GPU ray-marching through a 3D texture populated by the voxel data (Figure 5.15. This layer only requires a cube to be drawn, with front-face culling, and a shader that performs the ray-marching. Colors are applied for each voxel based on a colormap, and transparency is assigned using the luminance of another colormap.

5.3.4 Data Management

Before we began designing Artifact Based Rendering, several members of our team made extensive use of Paraview. Paraview is an application for creating 3D data visualizations, built on the Kitware's Visualization ToolKit (VTK) library. Paraview supports loading and extensive filtering of VTK datasets, and provides rendering controls for 3D point, line, surface, and volume data. Through the selection of colormaps and the application of several pre-defined geometric 3D glyphs, Paraview can produce 3D visualizations that follow commonly accepted data visualization theory (Figure 6.2 on Page 108). Recent versions of Paraview even allow visualizations to be viewed on VR head-mounted displays.

While Paraview has proven sufficient for creating accurate, perceptually-acceptable visualizations of most 3D datasets, we found that the images it produces lack the visual richness and expressiveness of tools like Visualization by Sketching [Schroeder and

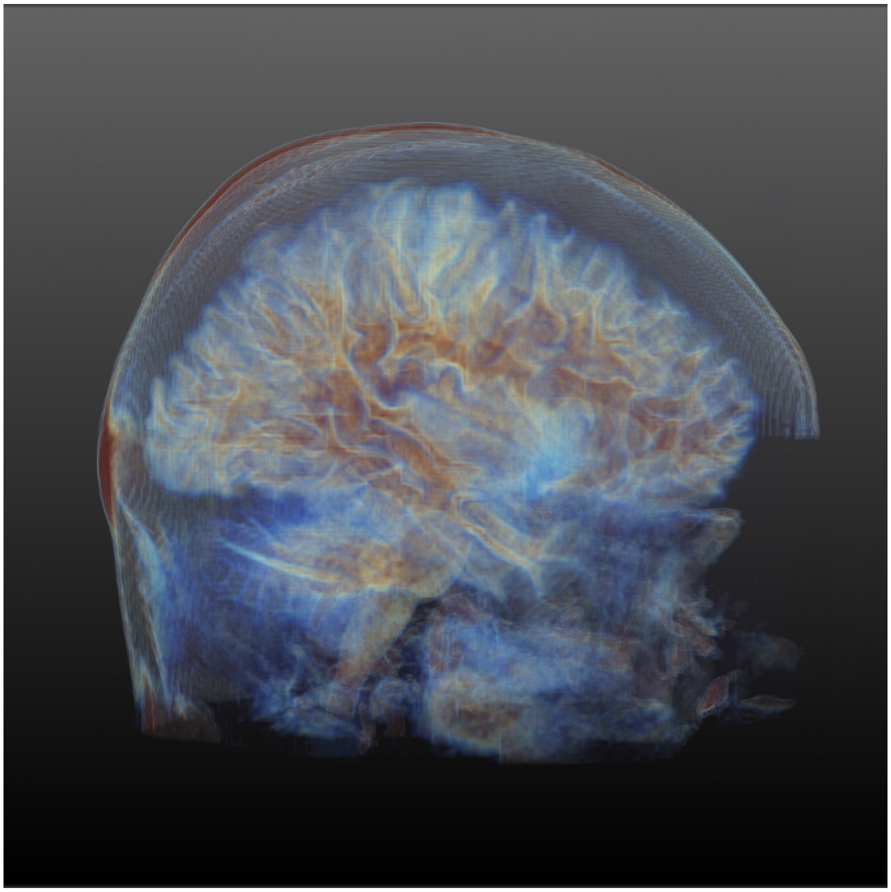


Figure 5.15: An example of volume rendering using a colormap as a transfer function to encode a volumetric variable.

Keefe 2016], with which our artist was equipped to produce highly expressive 2D visualizations. We observed that Paraview's rendering implementations do not support the visual variety that our artist could dream up, nor does the interface support an iterative creative design process conducive to exploring rich visual variation. For example, it provides no built-in means of importing custom glyphs, nor does it support texturing surfaces with images. Furthermore, Paraview's VR capabilities are currently limited and don't support any dynamic user interactions.

Thus, we decided to use Paraview for what we found it to be successful for – data wrangling and filtering – and build our own 3D rendering pipeline with a focus on data-driven visual variation and iterative design using Unity3D.

We selected Unity3D for its highly customizeable graphics rendering engine, and its out-of-the-box support for VR interactions across nearly all available VR head mounted displays. However, as a Game Engine, Unity3D is not automatically equipped to support complex 3D datasets. Our first task was to enable flexible Paraview-like data loading in the Unity3D's C#-based platform.

Creating a VTK plugin for Unity3D

VTK comes with a vast C++ and python API that handles data loading, parsing, and filtering. To allow Unity3D to take advantage of VTK's data processing power, we needed to produce a C# wrapper around the VTK library.

Using Python, we implemented a script that parses the VTK C++ header files and generates both C bindings and a C# API that matches the C++ VTK API. The script takes in a list of desired VTK class names, and pre-processes the respective C++ header files (to unfold the many pre-processor macros the VTK API relies on). Then the script constructs a symbol table for the classes and public methods through regular expressions. Finally, this symbol table is used to create plain-C bindings for all the many overloaded VTK methods, and a C# library that mimics the C++ VTK class hierarchy and method signatures.

Upon compiling the C-bindings into a .dll (along with a statically-linked build of the VTK library(and importing the generated C# scripts into Unity3D, Unity3D C# scripts can be developed with code that directly matches the VTK example code and documentation. Additional Unity3D-specific extension methods allow VTK methods such as `double* vtkDataSet::GetPoint()` to return a Unity3D `Vector3`, or `double* vtkDataSet::GetBounds()` to return a Unity3D `Bounds` object.

Finally, the .dll and C# scripts are exported from Unity3D as a `.unitypackage` and can be imported into any Unity3D application, even on a system that did not previously have VTK installed on it.

VTK for Artifact Based Rendering

VTK file formats are used to bring data from a wide range of sources into the rendering engine implemented in Unity3D. We use Paraview and Python interfaces to the underlying VTK toolkit to load raw structured and unstructured grids representing scalar, vector and tensor fields, filter the data to create geometrical representations, then write the results in forms immediately accessible to the visualization system. Figures in **??** include dataset slicing, isosurface extraction and streamline advection performed using Paraview and vtkpython.

We augmented the VTK toolkit to provide sampling methods that are key to our glyph representations of data. For volumetric data, we sample multi-dimensional probability distributions using a Metropolis-Hastings (MH) algorithm [Chib and Greenberg 1995], a Markov Chain Monte Carlo method. Our software supports a *transfer function* that maps the input data range of interest to a probability distribution. Metropolis-Hastings sampling from this mapped result then produces a set of samples distributed with a density proportionate to the user's interest.

We also have implemented several methods of sampling geometrical elements. We sample particle traces based both on equal steps in arc length and integration time. This enables us to present the local velocity either by distorting evenly placed glyphs or by irregularly placing undistorted glyphs. We also have implemented area-based random sampling of surfaces (independent of VTK's limited glyphing capabilities), enabling us to instance complex glyphs along the surface in the Unity visualization engine.

5.4 Discussion

5.4.1 ABR Design Guidelines

We can provide some preliminary design guidelines informed by our work with ABR thus far. At a high level, we note that the structure shown in Fig. 5.3 provides artists with artifact categories common to their visual language. Designing a visualization follows the same process one uses when laying out the structure of a painting; blocking out the underlying structure using line, shape and forms. Once the structure is in place, artists can draw upon existing pre-loaded artifacts or create new ones, naturally applying the principles of design (Balance, Repetition/Rhythm, Focal Point/Dominance, Unity/Category [Evans and Thomas 2008; Lauer and Pentar 2012]) to organize the compositions, direct attention, clarify hierarchy, highlight relationships, and create unity. At a lower level, we have found the following specific design considerations to be most important for working with ABR.

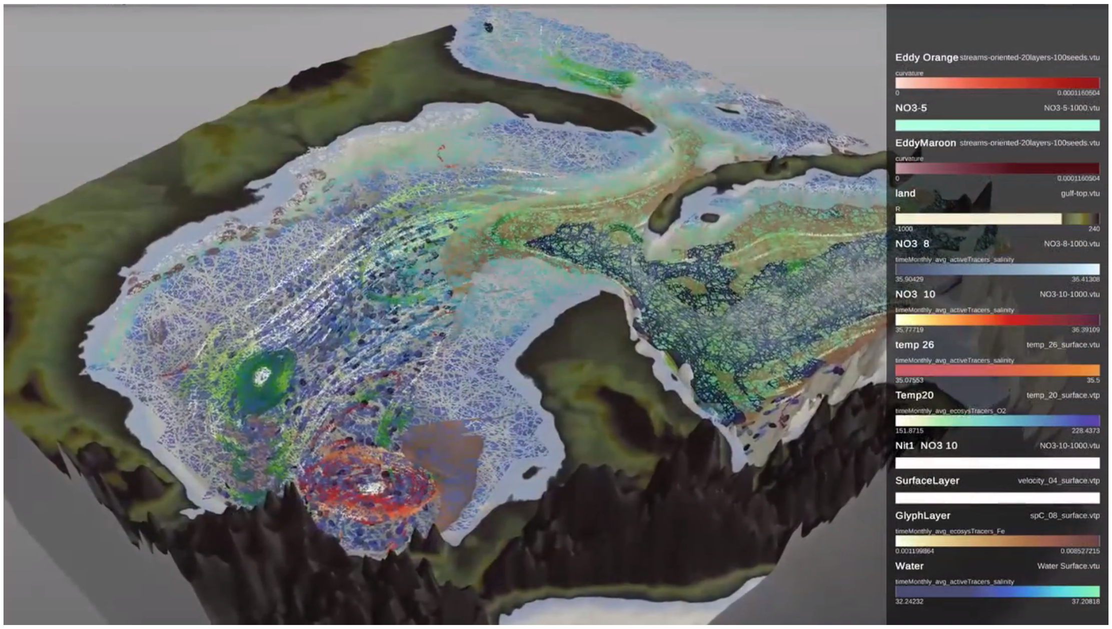

Eddy Orange streams-oriented-20layers-100seeds.vtu
curvature
0
0.0001160504
NO3-5 NO3-5-1000.vtu
EddyMaroon streams-oriented-20layers-100seeds.vtu
curvature
0
0.0001160504
land gulf-top.vtu
R
-1000
240
NO3 8 NO3-8-1000.vtu
timeMonthly_avg_activeTracers_salinity
35.90429
36.41308
NO3 10 NO3-10-1000.vtu
timeMonthly_avg_activeTracers_salinity
35.77719
36.39109
temp 26 temp_26_surface.vtu
timeMonthly_avg_activeTracers_salinity
35.07553
35.5
Temp20 temp_20_surface.vtp
timeMonthly_avg_ecosysTracers_O2
151.8715
228.4373
Nit1 NO3 10 NO3-10-1000.vtu
SurfaceLayer velocity_04_surface.vtp
GlyphLayer spC_08_surface.vtp
timeMonthly_avg_ecosysTracers_Fe
0.001199864
0.008527215
Water Water Surface.vtu
timeMonthly_avg_activeTracers_salinity
32.24232
37.20818

Figure 5.16: One of several results from the upcoming Chapter 6 ABR design study with the biogeochemistry data in the Gulf of Mexico [Wolfram et al. 2015]. The legend has been automatically generated from the colormaps used in the visualization.

Contrasting Forms. When choosing glyphs, consider contrast between forms. To create a visual contrast, pair glyphs that are: geometric versus handcrafted or organic, curvilinear versus angular, sparsely textured versus densely textured (e.g., the identity channel glyphs in Fig. 5.3 are arranged from dense to sparse texture).

Contrasting Profiles. Profiles of both glyphs and ribbons play a key role in visual distinction capability. Consider contrast in complexity, profile, and continuity (e.g., the streamlines in Figs. 3.1 and 6.1 (right)).

Contrasting Textures. Consider contrast between size and/or source of surface textures of objects. See Fig. 6.1 (left), for examples of glyphs with contrasting density and source of textures. As with forms, textures follow similar rules in creating contrast; likewise, some examples include organic versus geometric textures and curvilinear versus angular textures as shown in Fig. 5.11.

Color. The Color Loom applet enables quick construction of versatile colormaps. When selecting images from which to construct a colormap, select images that provide multiple types of contrast specifically: luminance; cool - warm; and low and high saturation. The range of contrast types will enable high-resolution; semantic associations; and highlighting capability. Ware and Rhyne provide in depth guidance based on these principals [Rhyne 2016; Ware et al. 2013].

5.4.2 Pairing with Perceptual Guidelines

Aside from giving artist users starting points informed by the perceptual literature, we have made an intentional decision to not constrain artists by encoding hard constraints or rules into the framework, instead favoring giving users the freedom to create and discover new visual encoding strategies that have not been explored before and may break some of the current rules. If successful, this means that ABR may lead to new visualization designs that researchers wish to verify and better understand using low-level perceptual studies. Thus, the two approaches are complementary. ABR might be thought of as a top-down approach to inventing and experimenting with new visual encoding strategies for visualization, whereas low-level perceptual studies of generic uses of color, shape, texture, and form might be thought of as a bottom-up approach. We are excited for these two research methodologies to continue to inform each other.

5.5 Conclusion

With Artifact Based Rendering, we present a new artist-accessible design process for leveraging the variation of the natural world and an artist's creativity to produce visualizations of complex multi-variate volumetric data visualizations. Visualizations created through ABR should enable deeper discernment of the data by encoding multiple variables in the same space simultaneously using multiple visual channels such as color, texture, and form. And these visual channels have unlimited variation and can be crafted to take advantage of natural associations we already have with different types of physical objects, making the data visualization engaging and accessible to a wide audience.

In this chapter, we examined the underlying theory and implementation of ABR. In the following chapter, we put ABR to the test by applying it to a number of different datasets. And by conducting a design study with an artist, we compare how ABR improves upon the workflows provided by more traditional visualization systems like Paraview.

Chapter 6

Applications and Results of Artifact-Based Rendering

6.1 Introduction

[1] In the last chapter, we took a deep look at the theoretical and technical structure of Artifact-Based Rendering, a framework of tools, algorithms, and processes that makes it possible to produce real, data-driven 3D scientific visualizations with a visual language derived entirely from colors, lines, textures, and forms created using traditional physical media or found in nature. In this chapter, we put ABR to work and show how we can apply our new techniques to a wide range of 3D scientific data.

To provide an initial evaluation of ABR, we report user feedback from some of these applications along with a within-the-research-team design study that characterizes how visualization design processes can differ with ABR as compared to traditional scientific visualization tools.

The main contributions of this chapter can be summarized as:

- Evaluation of impact on visualization design processes via a within-the-research-team design study comparing designing visualizations with ABR versus a traditional tool.

---

[1] This chapter is based on work published in IEEE Transactions on Visualization and Computer Graphics [Johnson et al. 2019b].

- Results and feedback for two domain science applications.
- Further examples of different datasets with ABR applied

6.2 Internal Exploratory Design Study

The quality of the work that results from design processes is one possible metric for evaluating ABR, and we investigate this with domain science collaborators in Section 6.3. However, having followed the proceedings of the BELIV workshop series that explores visualization evaluation techniques "beyond time and errors" [Sedlmair et al. 2018], we were motivated to consider more creative alternatives for evaluating impact on the visualization design more directly. Thus, the work in this section targets more direct metrics, such as the number and quality of design alternatives explored when faced with a real-world, challenging scientific visualization design problem.

Our long-term goal is to measure this type of impact on process in a larger-scale setting (e.g., building on recent workshops at IEEE VIS [Rogers et al. 2017, 2016] and the College Art Association Conference [Samsel and Keefe 2013]). The 2-day internal exploratory design study presented here is intended to be a first logical step toward this, providing an initial characterization and comparison of the design process with ABR vs Paraview, which serves as an example of a popular traditional scientific visualization tool, and piloting possible future evaluation methods and metrics.

6.2.1 Methodology

Given a new scientific dataset and time to discuss relevant data and research questions with a domain scientist, the **task** is to explore a variety of potentially useful visualizations for the problem. Since the design study is posed as an A vs B comparison, the artist performs that same visual design task first with ABR and then with Paraview. Learning effects are sure to be a factor, but, by scheduling ABR on day 1 and Paraview on day 2, the advantage falls to Paraview.

Our research team's artist, also the second author of the paper, acted as the only **participant** for the internal design study. Being a member of the research team, the artist had worked to co-develop ABR for more than 18 months. However, all of the features of the system were not ready until the week before the design study. Thus, her hands-on **training** creating art with traditional materials was a lifetime; her hands-on experience with the ABR toolset was less than 1 week; and her hands-on experience with Paraview was more than 5 years.

In a true workshop setting a formal introduction to the data would be required, but our case, the artist and scientist had already discussed high-level scientific goals as part of our ongoing collaborative project. The **dataset** comes from scientists studying mariculture, specifically commercially viable macroalgae growth. They have a challenging data analysis problem that requires understanding of many critical data variables from a fusion of remote sensing and a high-resolution computational simulation of ocean currents [Petersen et al. 2015; Ringler et al. 2013; Wolfram et al. 2015] extended to include biogeochemistry [Moore et al. 2001, 2013; Wang et al. 2014, 2015]. In offshore regions of the Gulf of Mexico, eddies in the ocean currents could provide "small farms" for macroalgae since the eddies can carry key nutrients, but to understand which eddies provide the most suitable conditions, it is necessary to relate eddy velocity, rotation, divergence, salinity, and temperature data together with nitrate, phosphorus, and other biogeochemical variables, a few of the 30 variables within the dataset – all within the context of the local geography. To facilitate direct comparison, the same data were utilized on day 1 and day 2.

We recorded the design process by saving state files and screenshots at regular intervals and whenever an interesting image was produced. We also logged time performing design functions (e.g., refining color palettes, sculpting glyphs, sketching).

6.2.2 Results and Interpretation

Figures 6.1 and 6.2 document process and results for the two tools. Since rapid experimentation and broad thinking is the foundation of all creative design processes [Buxton 2010], we also report data on the number of designs and design elements explored and time performing design activities. With ABR the artist designed with 72 line textures and glyph artifacts. 49 artifacts were created during the study, including 43 new hand-painted artifacts; 6 new hand-sculpted glyphs and 16 new colormaps.

The total time working with ABR was 7 hours, 46 minutes. 2 hours, 19 minutes were spent on Stage 1 of the ABR pipeline (crafting and making), with 8 minutes of

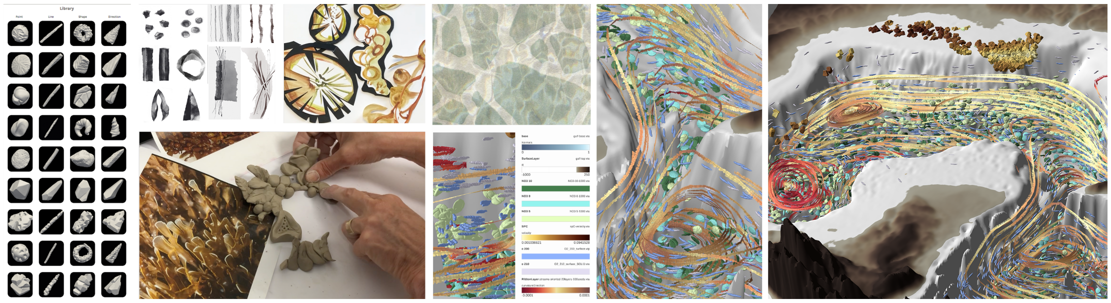


Figure 6.1: Process and results from the internal exploratory design study with ABR on the biogeochemistry data in the Gulf of Mexico [Wolfram et al. 2015], left to right: pre-made glyphs; glyphs painted during the study; glyphs constructed during the study; textures captured pre-study; detail of final visualization; visualization of the Gulf of Mexico.

that time devoted to searching for reference imagery online and the rest working with clay, paint, and wire. The total time spent on Stage 2 (digitizing and translating) was 3 hours, 25 minutes. The total time spent on Stage 3 (data-visual mapping and VR visualization) was 2 hours, 2 minutes.

With Paraview, 4 existing glyphs, testing a range of sizes, were explored from the 5 available built-in geometric primitives, and 14 existing colomaps (Figure 6.2). The time working with Paraview was shorter than expected going into the study; just 2 hours and 43 minutes.

The most interesting result is the difference in time and motivation devoted to design with each tool. Paraview includes just three non-directional glyphs, so the options for glyph combinations were quickly exhausted. If the artist had not already spent time exploring color on day one, more time could have been spent usefully exploring color in Paraview – this is the area in which artists are most able to contribute to design with Paraview. However, since she had already spent considerable time on color, she quickly reached a point of diminishing returns on day two; frustration due to limited options for glyph forms grew once she had balanced the size and color intensities of the nitrates, and she did not want to continue.

By contrast, with ABR, the artist worked over 7.5 hours to test the basic glyph and line design options. She reported that the power and versatility of ABR gave her a range of design options comparable to those available in her physical studio, where

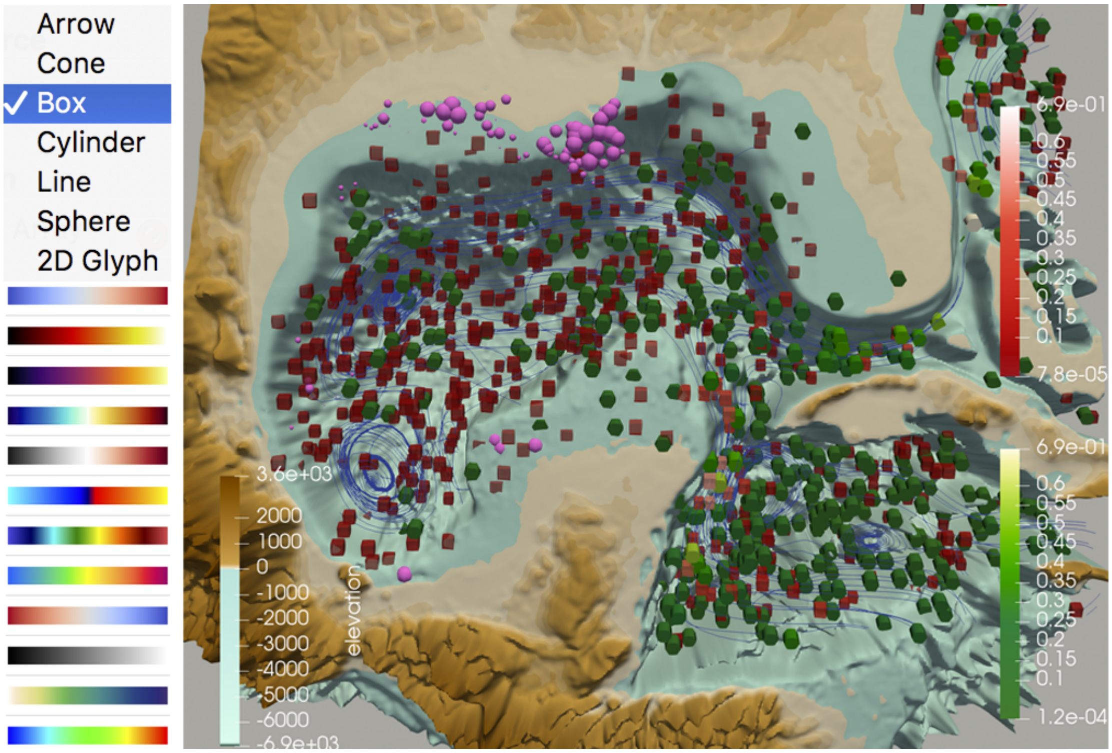


Figure 6.2: Left - Encoding options in Paraview, Right - Results from the internal exploratory design study with Paraview.

possibilities are essentially limitless. A key finding was that though it may seem that ABR would be a comparatively slow method of designing glyphs, in fact, the artist was able to iterate through many forms based on the needs of scientists with more speed and more precise results than she was able to with previously available tools. This mode of quick, preliminary iteration and experimentation was crucial in harnessing the power of the increased visual vocabulary that ABR enables. The process resulted in a broad range of possible solutions suitable to addressing the complex visualization problems posed by large, multivariate datasets and complicated scientific questions. While making individual forms was not in itself time-consuming, with such a capacious range of options for design, the artist did need to spend more time honing in on those best suited to the research questions and visualization needs.

Qualitatively and quantitatively, we can say that ABR enabled a broader exploration of the visualization design space. Shneiderman outlines requirements for computer-based creativity support tools as enabling results that are both novel and useful [Schneiderman 2007]. Simply comparing the process and result imagery in Figure 6.1 to typical VIS proceedings makes a clear case for novelty. Despite the limitations of this early study (a single impossible-to-be-unbiased user, an exploratory rather than scientifically controlled study), we believe these results already demonstrate that ABR is enabling us to reach and visually critique points in the scientific visualization design space that we have never explored before. The study does not specifically address usefulness. Thus, we take an early step toward this evaluation in the next section.

6.3 Applications and Guidelines

We have also evaluated ABR by applying the new framework together with collaborators on several actively researched scientific datasets.

6.3.1 Macroalgae in the Gulf of Mexico

The first application uses the same Gulf of Mexico data as the design study. This section reports on follow-on efforts with these data, taking a week of time to refine the visualization and seek feedback.

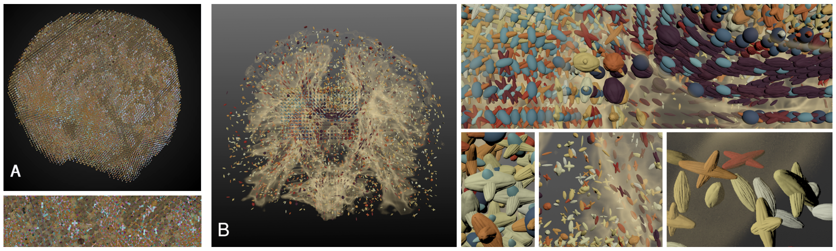
Figure 5.1 along with the accompanying video document the visualization results achieved using ABR. Color maps were generated to provide a natural color palette using artifacts from photographs and paintings. Eddies are shown using textured ribbons to depict the flow lines. Color encodes rotational direction (green-blue for cyclone, orange-red for anti-cyclone), and the texture itself is varied along each ribbon based upon the local degree of curvature. Temperature and salinity are encoded using textured surface layers. Three isosurfaces of temperature are shown at levels to evaluate macroalgae growth (20, 25, and 26 degrees Celsius). These are also colored and textured using an inkwash texture set; the texture is denser when salinity increases. Finally, three custom glyphs are distributed using density based sampling to depict three types of nitrates. The colors for the three were chosen to be analogous and also vary in response to the local salinity.

The collaborating domain scientist posed the challenge of being able to clearly visualize 5 or more variables simultaneously because he has found this impossible to accomplish using his current toolset consisting of Paraview (via slices, isosurfaces, and volume rendering) and python-matplotlib (to render final plots for publication after finding areas of interest via Paraview). Our interpretation is that these current tools are not visually expressive enough – one can only overlay so many surfaces and volume clouds before the different fields become too difficult to discern. Upon seeing the results in Figure 5.1, the scientist reported that ABR will be "transformational" to his science, saying these pictures can easily visualize more than five variables. With this type of glyph-based visualization, "you can superimpose functional relationships between multiple fields [such as nitrates getting caught in eddies and pulled to the surface] in a way that you can't with volume rendering or surfaces alone." Similarly the textured isosurfaces with color have the potential to encode "three in one", packing more data into the multivariate visualization. On the aesthetic, he commented: *They look like biology more than they look like plastic. They could be real, produced by nature. I think that people are going to underestimate that. At first, I'm perturbed, these don't look like plastic, then I realize this is not a problem but a major benefit.*

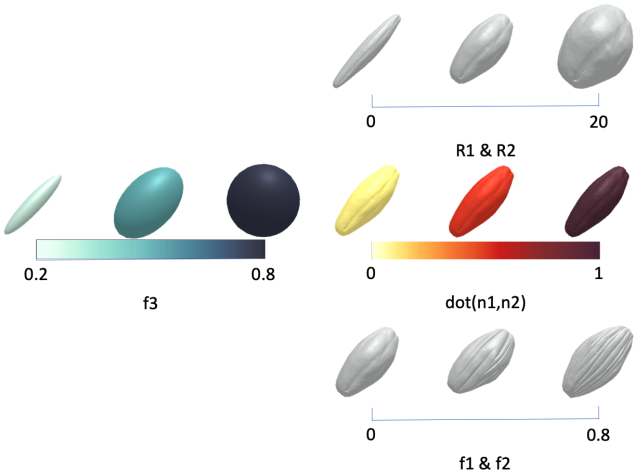
6.3.2 Brain Microstructure Imaging

In another application domain, scientists are developing computational tools to leverage high-field Magnetic Resonance Imaging (MRI) for understanding structural and functional alterations of brain connections in neurodegenerative disorders. The latest algorithms in this field make it possible to not only identify several crossing pathways in white matter areas with complex configurations but also to estimate microstructural parameters, such as axonal diameter and density [Farooq et al. 2016a,b], increasing the difficulty of visualizing Diffusion-Tensor MRI data, which is already regarded as a significant 3D visualization research challenge. To date, such data have only been visualized using slice-based approaches (e.g., [Farooq et al. 2016a]).

Figure 6.3(a) is a straightforward translation of the prior slice-based visualizations to a true 3D visualization. Clearly occlusion is a major factor in designing an effective 3D representation, and this straightforward translation is not successful. Section B shows a refined "magic lens" design. Volume rendering and sparsely sampled glyphs provide



(a) A: Straightforward extension on slice-based methods to 3D. B: ABR designed visualization with high-resolution data in the centrum semiovale.



(b) A custom legend showing the mapping of glyph attributes to the multiple variables in the ABR visualization above.

Figure 6.3: Visualizing brain microstructure in 3D.

context throughout the brain, and high-resolution data are presented in an interactively defined small volume. The voxels are filtered to display only regions with crossing fibers using oriented glyphs to show the primary and secondary fiber orientations. A glyph set was designed so that each glyph has a similar profile, but the density of the linear texture along the length of each glyph increases in response to the axonal density variable. The glyphs are also sized based on the axonal radius parameter. An oriented ellipsoid depicts the “leftover” diffusion for each voxel after computing the two primary fiber orientations, and a volume rendering provides anatomical context.

The collaborating domain scientist suggested a focus on the centrum semiovale (highlighted in Figure 6.3(b)), which is a region with high fiber crossings. From the primary and secondary fiber orientations, the visualization confirms expected brain structure in this region. This is the first time the scientist had seen the data in a true 3D display, and looking at the visualization in VR convinced him that he can see structure that is not possible to see in 2D slice-based visualizations. Similar to the Gulf of Mexico results, the aesthetic produced with ABR leads to a natural, organic visual language, and by encoding data with textural variations of the glyph rather than color (as used in prior slice-based approaches), we were able to use color to encode a new derived variable (similarity between primary and secondary directions), which, with a color map applied, calls visual attention to crossings that are nearly perpendicular (red color range).

6.3.3 Astrophysics

During the early stages of the evaluation of the design foundations, library, and software systems, we applied these to visualize an astrophysics simulation. Datasets were provided by collaborators whose research couples a high-resolution hydrodynamics simulation of the early universe together with a hydroxyl and water-producing chemistry model in order to determine how water molecules would be created and distributed in space and time in the early universe [Wiggins and Smidt 2018]. By comparing simulation results to astronomical observations, the scientists aim to verify both the hydrodynamic and chemical models of ancient water formation.

Figure 6.4 shows two views and possible visualizations of the simulation. Here, the task scientists need to perform is to track the concentrations of three heavy metals needed for water formation (CH4, CO, OH) and relate these to water formation. The top two images are depicting close-ups of the water (blue crescents) with the density volume rendered in light blue, and the metals represented in white disks of high angular texture, referencing rock and metal. Understanding the relationships between these variables can elucidate the necessary conditions for water formation, which can help to establish where else water may have formed.

Prior to these new visualizations, the scientists have only been able to see the variables depicted in the single 3D visualizations produced here by comparing six separate slices through the data or, more recently, a volume visualizations with three volumetric layers. The multiple volume layer approach at least provides three-dimensional context, and scientists were excited and gleaned new information from the data as a result of

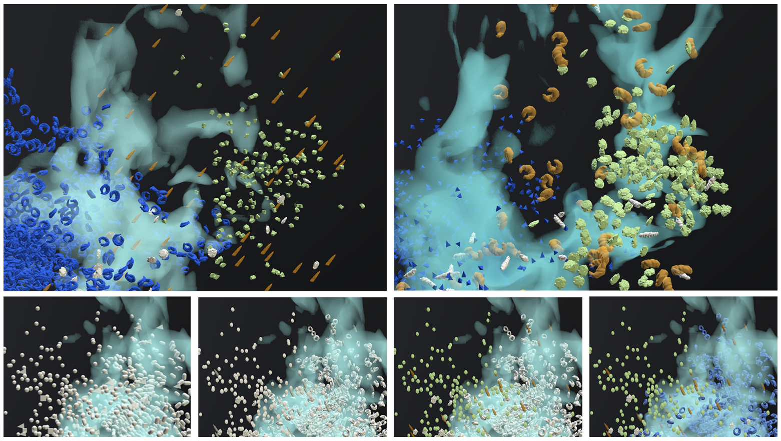


Figure 6.4: Top row: Two variations of handcrafted multivariate volume visualization simulating water formation in the early universe. The turquoise volume rendering represents the particulate density, the water is shown in the curvilinear flowing blue forms, selected for their detail and thus becoming a focal point [Lauer and Pentar 2012]. The orange, white and green represent the three heavy metals being tracked, CH4, CO and OH respectively. The comparison shows the artists iterations needed to accurately depict the science given the complexity of variables glyphs and modifiers within the system. The right side is the earlier image. The left, the final, providing the emphasis on the water, encoded with an associative form. Bottom row: This Snapshots from the design process of assigning *elements*, *families*, and *modifiers* to the data, starting with on the left with a geometrical family typical of present-day scientific visualization, moving toward a richer, handcrafted visual language, and then using color to double-encode the type of each variable.

seeing these three variables together in a single view [Wiggins et al. 2019]. However, we know that rendering multiple data layers as colored volumes obscures the data and leads to a need for the scientists to interpret complex color combinations.

Figure 6.5 shows a comparison of one of these original three-layer volume renderings (left) and a result from the new method (right). It's worth noting that an artist was also involved in creating the volume rendering on the left, and the effective use of color visible there took two days of hand-tuning volume rendering transfer functions in order to reach. One the right, the visualization includes six (twice as many) volumetric variables and the results are more clearly legible. Three heavy metals (CH4, CO, and OH) are rendered in warm tones on angular glyphs. The glyphs come from different families and are also

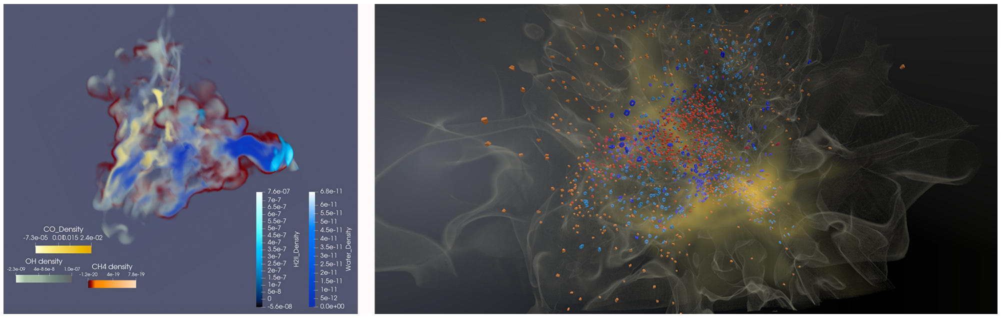


Figure 6.5: A supernova midway through its explosion. Left: A three-layer volume rendering. In related work, scientists used this visualization to for the first time see three of the variables from their simulation visualized in the same three-dimensional space. Right: A visualization using hand-sculpted families of design elements together with a single layer of volume visualization depicts twice the number of variables and uses the extended visual language to support hierarchical associations (e.g., the two styles of water glyphs are both round and smooth, the three styles of metal glyphs are all angular).

different visual elements. This provides the maximum visual distinction, reinforcing the color categorization. The precursor to water H2ii and the water itself are encoded with curved glyphs, referencing flow and thus having associated properties with water rather than metal. The yellow volume rendering of just a single variable depicts pressure.

6.3.4 Abstract Data and Future Work

There are many other possible applications that remain to be tested, such as more abstract data that do not exist within a predefined 3D structure. We believe the approach will translate well to abstract data because artists design visuals to convey abstract concepts, like human emotion, all of the time. It would be fascinating, for example, to ask artists to craft a series of glyphs to represent generic uncertainty, correlation and anti-correlation, or cause and effect. Future work also includes experimentation with new vis layers suggested by artist users and combining the handcrafted, organic aesthetic that is possible with ABR with more traditional geometric aesthetics.

6.4 Conclusion

The internal design study and applications are by no means a final evaluation; however, they clearly demonstrate an ability for the first time to create complete scientifically useful multivariate VR data visualizations using a visual language derived entirely from traditional physical media. The resulting aesthetic is novel, looking, as one domain scientist reported, like it could be “produced by nature”; we believe this has powerful implications for making science more understandable and engaging. Further, by leveraging skills artists already have with physical media, we believe ABR can have a powerful positive impact on the visualization community by broadening the diversity of people who can now contribute to creating 3D scientific visualizations.

The human-in-the-loop design process of ABR has the advantage of creating a pathway for artists, our society’s most accomplished visual thinkers, to engage in the scientific process. (This previously has been shown useful for creating data-driven 2D visualizations in a recent IEEE SciVis “best paper” [Schroeder and Keefe 2016], but not in 3D until now.) Artist involvement also naturally leads to visualizations with a decidedly different, more natural, hand-crafted aesthetic. Our experience is that when a viewer is “present” with the resulting data visualization in a VR environment, this different, natural, physical aesthetic is immediately obvious and even “felt”. Throughout this chapter, we saw still, 2D images of the results of ABR; but with a small stretch of the imagination, the remarkable experience of sharing a 3D space with these artist-driven visualizations may be appreciated.

Thus, we offer the results of Artifact Based Rendering as a primary example of Palpable Visualizations. We foresee many possible extensions to our technique, such as integrating many of the techniques described in the related work of the previous chapter. Work is already being done developing a visualization design and exploration user interface beyond the current Unity3D editor panels that we’ve implemented. These improvements and others can increase the palpability of scientific visualization along both the discernible and the accessible axes.

In the following chapter, we push the accessibility of our visualizations even further by taking strides to present ABR visualizations on affordable, wireless AR and VR headsets.

Chapter 7

Data Streaming and Remote Rendering for 3D Scientific Visualization

7.1 Introduction

We are living in an age of increasing remote collaboration and teleconferencing. For several years, services like Google Docs, GitHub, and DropBox allow researchers to work both asynchronously across the globe, and applications like Skype and Zoom have enabled synchronous collaborations to exist between diverse groups that may never have existed otherwise. While none of these tools can perfectly mimic the synergy of shoulder-to-shoulder teamwork or face-to-face discussion, the massive potential for more mutually beneficial interactions across distances outweighs the qualitative sacrifices of virtual technology.

However, we've repeatedly encountered one particular hole in this web of virtual collaboration tools during our own work on advancing 3D data visualization. Countless times we've been providing verbal updates to our teams of artists, scientists, and engineers, and have had to disclaim "...but to fully understand these advancements you really need to see it in VR," an experience which often involves either traveling across the country to visit our research lab, or purchasing a high-performance computer and a head-mounted display and learning the steps of accessing the relevant datasets and configuring our software for their system. Ironically, virtual collaboration has failed us in the area of sharing progress of virtual reality data visualization.

This obstacle is not limited to sharing AR/VR progress with fellow collaborators. We've encountered at least three types of limitations around the issue of geographically-stationary resources.

1. As outlined in the previous paragraphs, the difficulty of sharing progress of AR/VR-based work with fellow researchers can hinder advancements in techniques and applications due to lack of shared experience.
2. The datasets being visualized are becoming larger and more dynamic, and it can be time-intensive to curate and transfer relevant subsets from supercomputers in Texas to computer graphics research labs in Minnesota so real-time AR/VR applications can access the data locally.
3. In order to share new AR/VR-related data visualization advancements with the public, the public either needs to travel to the AR/VR research lab, or else expensive hardware needs to be transported carefully and undergo sometimes hours of set-up for a guided public demonstration.

But at the same time as our scientific data sets and our rendering techniques get more complex, low-end VR displays are becoming more widely available. Soon, it won't be inconceivable that a classroom of children may have access to 1:1 personal VR or AR headsets just as they now may have access to a personal tablet computer. And most science collaborators will probably have some degree of access to one of these low-end AR/VR displays, even when they're collaborating remotely. This increasing ubiquitousness of affordable immersive hardware raises the question of how they might support the sharing of scientific data visualizations for explaining science to the public or discussing research with collaborators.

While these low-end, affordable untethered devices may have high-resolution VR displays and tracking (like the oculus Quest) or advanced Augmented Reality technology (like the Magic Leap), they have dramatically lower storage, computational, and rendering capabilities than a VR display connected to a expensive high-end research

PC. Usually these low-end displays run on a completely different architecture and operating systems than the Windows and Linux machines on which so much immersive data visualization research is developed.

Analogously, even the traditional high-performance PC's used for high-end virtual reality data visualization techniques are not able to store the entirety of a dataset being analyzed. Sometimes this is because the datasets are simply too large - some simulated time-series can be petabytes of data. And sometimes this is because the data is being simulated on-the-fly, based on new input and questions from the researcher as they study the visualization. These simulated datasets are pushing the limits of massive supercomputers, and often only a small static subset of the data is copied to a PC being used for rendering an immersive data visualization.

Must supercomputer-scale datasets be reduced in fidelity or scope in order to be visualized with PC-based immersive visualization design workflows? And should visualizations be designed, optimized, and even limited to support a wide range of affordable hardware? Or are there techniques that free up visualizations to be driven by massive datasets, and be as rich and expressive as designers can make them, while still being accessible on any VR/AR device?

In this chapter we will present our new techniques for streaming data from supercomputers to PC-based immersive data visualizations, remotely rendering scientific visualization content for low-end affordable AR/VR displays, and supporting synchronous remote collaboration around a virtual data visualization. We specifically apply these methods to flexible Drag-And-Drop Unity plugins and demonstrate them with visualizations designed through Artifact-Based Rendering.

The contributions of this chapter are as follows:

- An architecture and demonstration of real-time data streaming from a supercomputer to a consumer laptop running an interactive visualization
- An architecture and demonstration of collaborative remote rendering from a high-performance rendering computer to consumer untethered VR & AR displays with parallax correction techniques
- A characterization of the performance of data streaming and remote rendering for accessible data visualization

- An analysis of the most fruitful opportunities for future work in Remote Rendering

7.2 Related Work

This work follows in the footsteps of previous research and development in both visualizing pieces of large datasets, and remotely rendering for untethered head-mounted displays. Here we visit several of the most relevant prior works upon which our work builds, or from which our work diverges.

7.2.1 Remotely Visualizing Large Volumetric Datasets

Often, volumetric datasets can take more memory than a desktop computer can handle, with simulations sometimes consisting of petabytes of numeric data. Producing an interactive visualization of data like this requires judicious sampling strategies and carefully engineered rendering techniques.

A straight-forward approach to remote visualization is to stream entire images from a server-side visualization tool to a client computer. In 1999, Engel et al. demonstrated early techniques for rendering entire images on the server side and streaming the pixels to a client [1999]. To improve transmission speeds, various approaches to compression were considered, such as Friesen and Tarman's 2000 approach of using dedicated hardware to convert RGB images to NTSC video and transmit 30 fps video at 1280 x 1024 [2000]. Much more recently, in 2017 Raji et al. demonstrated how volumetric scientific data can be remote-rendered from the cloud for embedding in web pages [2017], and in 2018 they extended this to allow remote-rendering into a Microsoft HoloLens AR display with head-tracking [2018]. However, latency was a major issue, and they reported 4 frame-per-second refresh rates, unacceptable for comfortable immersive viewing.

As we consider applying image streaming for viewing through untethered AR devices like the HoloLens, Magic Leap, or VR displays like the Oculus Quest, we are held to a higher standard for responsive framerates, as explained by LaViola Jr [2000]. According to LaViola Jr, Cybersickness is primarily caused by discrepancies between head movement and the perceived image in the display. One way to avoid this type of cybersickness might be to capture 360-degree images of the visualization and stream these to the user (such as Qian et al.'s technique [2018], referenced in the following section). However, this approach does not provide true stereo-scopic depth cues nor depth cues from head translations, both of which Aygar, Ware, and Rogers argue are important tools for understanding the 3D structure of such datasets as particle-based simulations [2018]. Thus, we look towards remote-rendering methods that allow for low-latency stereo examinations.

In 2002, Luke and Hansen presented the Semotus Visum framework to integrate several remote-rendering approaches [2002]. They describe three rendering approaches that are all integrated into their approach: Image Rendering, the technique described above where all the rendering is done on the Server Side; Geometry Rendering, where the server is responsible for sampling the data and producing sets of polygons that can be rendered on the client; and ZTex Rendering – also referred to as Depth Image Based Rendering (DIBR), which renders both a color texture and a depth map of a visualization from a particular view on the server, and then renders both on the client side as a textured grid distorted by the depth map.

Geometry Rendering is a useful technique for taking a complex volumetric dataset on a server, and visualizing iso-surfaces, particles, and streamlines on the client side. For example, Engle et al. built several web-based volume data viewers that transmitted iso-surfaces to allow the user to request iso-surfaces of varying iso-value and quality and have the mesh streamed to the client where it could be rendered in real-time [1999; 1998]. The sampling and production of these sorts of geometry can be easily performed server-side on a diverse range of datasets by Paraview [Ahrens et al. 2005]. And the local visualization of these geometry objects can be done on a modern PC using many visualization approaches, such as Artifact Based Rendering [Johnson et al. 2019b]. One warning to note about Geometry Rendering is that there is no upper bound on the size or complexity of the geometry being sent. It can be generally thought of as a subset of the original volumetric data, but may still tax the memory and rendering capabilities of the client computer.

DIBR takes its inspiration from depth-based rendering optimizations. Relief mapping [Oliveira et al. 2000] and Parallax occlusion mapping [Tatarchuk 2005] are two such methods for taking a gray-scale height map of complex geometry and computing-per-fragment what shading would appear were the surface distorted according to the height map. Displacement mapping [Heckbert 1986] takes similar inputs, but instead actually distorts the vertices of a mesh according to the values sampled from the height map. Lukasczyk et al.'s Voidga demonstrates a process for using DIBR to capture important views of large spatial datasets using depth information to reconstruct 3D geometry that provides a reasonable view that can be rendered interactively in a manner similar to displacement mapping [Lukasczyk et al. 2018]. An advantage to Displacement mapping-based DIBR over Geometry Rendering for remote rendering is that there is a bound on rendering requirements, as the actual geometry being rendered is a pre-computed gridded mesh. This ensures that any given frame in an untethered HMD can be rendered at a consistent framerate even as the head position changes. A disadvantage is that there can be holes in areas originally occluded from the viewer. Voidga demonstrates a partial fix for this by capturing views from multiple angles and joining the resulting meshes to fill in some of these missing regions.

7.2.2 Untethered HMDs, Remote Rendering, and Latency Mitigation for AR/VR

In 2011, Suma Rosenberg's team at USC introduced the world's first smartphone-based head-mounted display prototype and developed the FOV2GO, an open-source foldable immersive viewer that was the precursor to the Google Cardboard [Olson et al. 2011]. This provided a first step toward affordable, untethered VR displays based on mobile phone hardware such as GearVR, Oculus Go, and most recently Oculus Quest. In recent years, untethered AR displays have also been released, such as the Microsoft HoloLens and the Magic Leap, which – while not quite having reached the consumer market – indicate a future in which mobile immersive graphics will be a more and more ubiquitous technology. All of these devices provide a dramatic increase in accessibility for immersive content, but also come with severe limitations in both storage and rendering hardware as compared to tethered, PC-based VR systems (PCVR). To make up for these limitations, work has been done to stream rendered content onto these devices from remote servers based on real-time tracking data from the untethered devices.

One solution is to focus on spherical pre-rendered or video content. Quain et al. has worked with streaming high-resolution 360-degree video to phone-based displays by predicting head orientation and transmitting only sub-sections of the video feed [2018].

Techniques such as this, however, don't allow free exploration of an immersive environment or even full 360-degree stereo 3D imagery.

Several projects for generalized tracked streaming VR experiences are currently available as of 2019, such as Virtual Desktop for the Oculus Quest [virtualdesktop 2019] and the open-source ALVR [Polygraphene 2019]. These tools provide a means for streaming tracking data from the untethered device to a PCVR-capable machine, which renders the VR imagery before streaming it back to the untethered device. While these techniques mitigate latency in head orientation tracking through device-integrated 3 degree-of-freedom Asynchronous reprojection, they do not account for latency in head position.

This concept of "Asynchronous Reprojection" shows up in the hardware implementations of most consumer VR devices even when remote rendering is not involved, and is discussed by Google in their guidelines for VR development:

> Asynchronous reprojection has three major effects:
>
> - It ensures that the frame rate experienced by the user remains high, which is critical to user comfort in VR.
> - It reuses frames when the application isn't able to draw.
> - It ensures that movement within the app feels smooth and does not judder.
>
> Asynchronous reprojection is designed for 3DoF head movement, as it only corrects for rotational movement.
>
> Asynchronous reprojection does not correct for positional movement in 6DoF apps, which means if the app framerate drops the user will begin to experience a disconnect between the rendered and actual position. The lower the app frame rate, the larger the disconnect. Lower frame rates can also introduce errors associated with animations and other visual effects.
>
> (`https://developers.google.com/vr/discover/async-reprojection`)

Another approach that can reduce latency in remote rendering is split rendering, an approach taken by Lai et al. in their design study of remote rendering [2019]. They describe split rendering as exploring the key insight about immersive interactive applications that near-objects and far-objects have “contrasting predictability and rendering workload.” Their split-rendering approach, which they call “Cooperative Rendering,” is to separate the rendering of the background into a panorama that can be updated less frequently than foreground elements. They than use the local phone-based GPU to render foreground interactions and use the remote rendering server to produce periodic background panoramas which are then combined on the untethered display.

Also in November of 2019, Oculus released a feature called “Oculus Link” which provides an implementation of remote rendering that streams data from a PCVR-capeable machine to an otherwise untethered Oculus Quest via a USB C cable. During a keynote address at the Oculus Connect conference, Oculus CTO John Carmack explained several technical limitations that prevented Oculus from releasing over-the-network untethered remote rendering for the Oculus Quest [Carmack 2019], such as the unreliability of a WiFi-network to provide consistent frame updates. One option addressed was a split-rendering approach that would allow dynamic objects to be rendered locally on the untethered device while static objects would be rendered less frequently, but that for the foreseeable future, forcing developers to engineer their applications with a split-rendering approach was an unacceptable constraint. And even with their cable-based solution, Carmack admitted that only the head orientation is fully satisfactory, and that spatial translation of the head or hands still produces around a frame of latency.

Back in 2013, John Carmack described in great detail several possible latency mitigation strategies for VR [Carmack 2013], one of which is extremely similar to the Depth Image Based Rendering (DIBR) techniques by Luke et al. and Lukasczyk et al. [Lukasczyk et al. 2018; Luke and Hansen 2002] in the previous section. Abrash describes a generalized technique for forward-warping, where the framebuffer can be treated as a height field, depositing source pixels in their new positions after a head translation. He also describes the specific types of artifacts such as silhouette edges where “internal parallax would have revealed surfaces not visible in the original rendering.” These strategies were offered primarily for local latency on a tethered HMD, but at the end of the article he suggests a potential application for remote rendering where warping occurs on the client, but warns that the maximum amount of mitigated latency should be 30 or 40 milliseconds to avoid silhouette artifacts as much as possible.

We can see in these works that remote rendering is not yet a solved problem, especially in the area of translations, where simply pre-rendering a panorama that can be asynchronously reprojected does not produce the parallax effect our visual system relies upon for depth perception. We see this as a primary area to focus in our work on remote rendering.

7.3 Architecture

7.3.1 Motivation and Goals

In the effort to present our processing-intensive Artifact-Based Rendering visualizations on affordable, untethered head-mounted displays, we first turned to currently available remote-rendering solutions like ALVR [Polygraphene 2019]. ALVR was designed to stream PC VR applications and video games to the many types of affordable VR displays offered by Oculus, such as the 3-degree-of-freedom Oculus Go and Gear VR, or the 6-degree-of-freedom Oculus Quest. Out-of-the-box, ALVR provided what at first appeared to be a passable head-tracked, stereoscopic view of our Unity ABR visualizations. However, upon sharing the experience with other collaborators, we discovered that ALVR the remote-rendered experience induced Cybersickness for several of our team members who otherwise had little trouble exploring visualizations rendered locally.

Upon carefully examining the characteristics of ALVR that led to cybersickness, we consistently noticed that symptoms occurred when making translational head movements. As we examined the implementation of ALVR, we learned that ALVR leverages the devices integrated 3-degree-of-freedom reprojection to mitigate head rotation error caused by the natural delay of the two-way communication between the device and the rendering PC. As expected, even with a high-end dedicated wireless router, ALVR reported that we were getting latency of 50-80 milliseconds from the capturing of head-tracking data to the display of the respective image for the user. For comparison, a locally-rendered VR experience is expected to have a motion-to-photon delay within one render frame, like 11.11ms for a 90Hz display to 8.33ms for a 120Hz display. So when ALVR receives a remotely-rendered frame, it has already been at least 5 times longer since the user's head position has been captured than on a tethered headset.

ALVR can adjust for mild rotational offsets by spherical reprojection, but if the head has translated in the meantime, the expected parallax between foreground and background objects is delayed. It's this motion-to-photon delay that we hold accountable for inducing cybersickness in our team members [LaViola Jr 2000].

Rather than focus on solving the generalized case of remotely visualizing any VR content on an untethered AR/VR device (as ALVR attempts to do), we decided to focus on providing a non-cybersickness-inducing experience for a specific 3D data visualization use case from our experience working on the applications described in the previous Artifact-Based Rendering discussion. In such cases, the datasets are simulated on a supercomputer, and subsets of the data are manually copied to a desktop PC, where Artifact-Based Rendering techniques are applied, and (thus far) the results can be displayed on a local, tethered VR display. The specific visualizations we are concerned with are composed of surfaces, ribbons, and glyphs, which are colored and textured through Artifact-Based Rendering techniques. These elements combined often consist of over a million vertices during a given frame, exceeding the rendering capabilities of untethered AR/VR displays like the Magic Leap or the Oculus Quest (which may begin to drop below 60 fps after 250,000 vertices [Oculus [n.d.]]). As described in the introduction of this chapter, we see great advantages to faster data synchronization between the supercomputer and ABR-powered desktop PC for the sake of dynamic exploration of multiple timesteps, and we see great advantages to accessing Artifact-Based Rendering driven visualizations on multiple remote, low-cost, untethered AR and VR displays simultaneously for the purposes of remote collaboration and engagement with the public. Our goals in this research are as follows:

- Dynamically synchronize data objects between supercomputer simulations and desktop PC's producing Artifact-Based Rendering Glyph and Surface visualizations
- Display stereoscopic, head-tracked views of the ABR Glyph and Surface visualization on remote Magic Leaps and Oculus Quests
- Avoid cybersickness due to instantaneous discrepancies between head movement and the perceived image in the display [LaViola Jr 2000]

- Provide accurate spatial placement of 3D structures as perceived through depth cues from stereo vision and head translation [Aygar et al. 2018]
- Facilitate multiple simultaneous remote viewers
- Open-source an extensible Unity-based remote-rendering framework for incorporating future techniques for new applications
- Support drag-and-drop, out-of-the-box remote rendering for arbitrary Unity-based projects, allowing for further refinement and optimization through developer-specified settings

It is worth noting that we are not trying to produce a flawless representation of the original ABR visualization on the remote devices. As our current goals are not to maximize presence (differing from those of the video gaming industry), but instead to maximize comfortable and structurally accurate views, we will tolerate temporary negative visual artifacts such as the silhouetting described by Carmack [2013].

7.3.2 High-Level Architecture Design

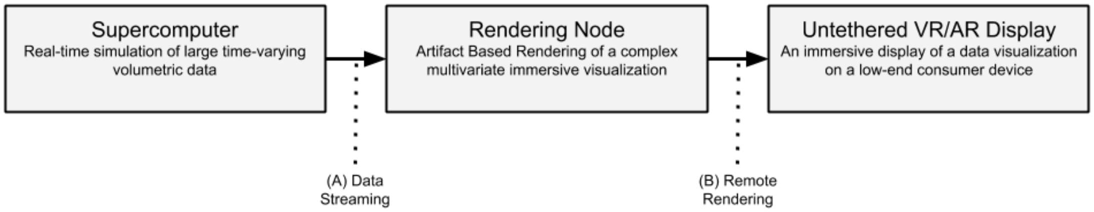

Supercomputer
Real-time simulation of large time-varying volumetric data
Rendering Node
Artifact Based Rendering of a complex multivariate immersive visualization
Untethered VR/AR Display
An immersive display of a data visualization on a low-end consumer device
(A) Data Streaming
(B) Remote Rendering

Figure 7.1: Three connected nodes - A supercomputer, a high-performance graphics computer, and a consumer AR/VR device - are connected by two types of transmission strategies: A) Streaming simulated data from the supercomputer onto a rendering node, and B) Streaming a pre-rendered immersive view from the rendering node to the low-cost HMD.

We approach our goals in two stages. Figure 7.1 Shows the interactions between three pieces of computational hardware. This approach separates the supercomputer containing the large simulation data from the client user's HMD by a rendering node. Thus, to go from the simulation data to the HMD by way of Artifact Based Rendering, there are two separate transmission stages: Data Streaming and Remote Rendering.

Data Streaming

Since many complex datasets (like the bio-geochemistry described in Chapter 6) are comprised of multiple terabytes of volumetric data, the data is sampled into geometry like iso-surfaces and streamlines which are transferred to a VR-capable computer. Then visualizations are designed around these subsets of the data. This limits the visualizations to only displaying what can be stored locally on the rendering computer, making it difficult to dynamically re-sample the data on-demand (with new streamline seed points, for example), or switch between time steps.

Data Streaming would entail replacing these direct references to local files with network-connections to a dataset server on a supercomputer. This server could re-sample the data on-demand based on either requests from the ABR visualization system or local updates to the simulation data, sending sampled geometry with desired timesteps or sampling parameters in response. When the transmission is complete, the rendering node would re-generate the visualization based on the newly received data.

Remote Rendering

In addition to the space limitations, low-cost untethered AR/VR displays also have major limitations in graphics rendering power. They may simply not be able to process tens of thousands of glyphs or ray-tracing or other advanced rendering techniques that are commonplace for scientific data visualization. Normally, this limits the visual fidelity of immersive experiences on these low-cost untethered displays.

Remote Rendering off-loads these graphics-intensive tasks to a larger, high-performance rendering PC and transmits rendered imagery to the client's untethered display. This requires the client display to regularly update the rendering node with its latest tracking data, and then imagery is rendered from the requested viewpoint and transmitted back to the display. The display then performs corrections for any tracking error introduced during the transmission delay.

7.3.3 High-Level Implementation Approach

While supercomputer simulations, visualization tools, and untethered AR/VR clients may be written in many different languages, we chose to focus on implementations where the data server is running Paraview, the rendering node's visualization tool is a Unity application, and the untethered AR/VR client is also running a Unity application as a viewer.

We chose to focus on Paraview for the data server because it's a common tool for many of our collaborators for working with simulated large datasets, and is also easily accessible on Desktop PC's for rapid development and testing.

We chose to focus on Unity for the rendering node both because our Artifact Based Rendering implementation is built on Unity, and also because Unity is a widely accessible platform for other visualization research projects.

And we chose to again focus on Unity for the client viewer as it provides support for all of our target untethered AR/VR displays. With Unity, we can easily deploy and debug the same application on the Magic Leap and the Oculus Quest.

We've designed a python-based ParaView filter that can send any number of VTK data objects to a remote computer anytime a timestep or filtering parameter is changed, along with a synchronizing message that allows the Unity3D application to update its visuals after all affected geometry has been replaced by the data stream. Figure 7.2 shows a sequence diagram of the following communication. The Unity3D application running our ABR visualization system was augmented to connect to the supercomputer via TCP sockets, replacing our original code that relied on VTK files stored locally on a computer. Whenever ParaView had new data to transmit, the Unity3D application asynchronously collected the new VTK data objects via the TCP sockets, producing new renderable geometry in the background as a type of “back-buffer”. Once the synchronization message signals the new data objects have all been sent, the Unity3D application replaces the out-dated geometry with the new streamlines and glyph fields, resulting in a synchronized update showing the latest data-driven objects. This way, the rendering laptop never needs to store any of the data on its harddrive, and instead only stores at most two timesteps worth of dynamic data in memory at any time.

This approach also works with non-supercomputers, and can even run with a local instance of Paraview on the rendering computer. Even without using a supercomputer, this live connection with Paraview enables interactive manipulation of the data sampling on Paraview while the ABR visualization automatically updates every time the dataset changes.

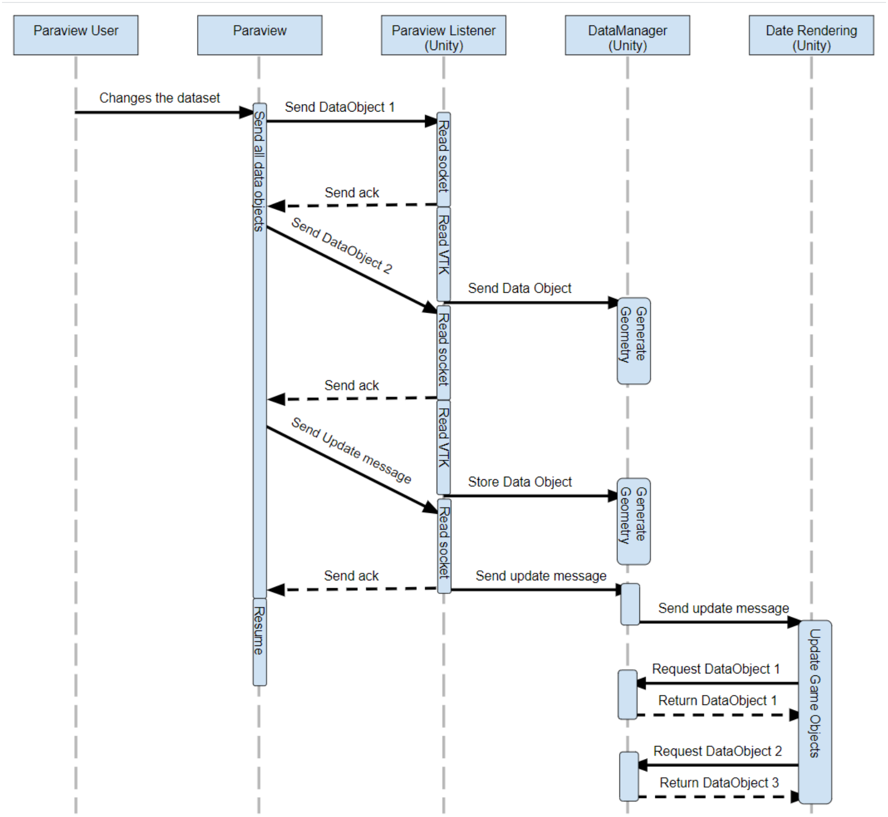


Figure 7.2: A sequence diagram showing how two data objects are transmitted from a machine running Paraview to a machine running Unity upon some change to the dataset (e.g. an incremented timestep or a new sampling parameterization).

Our approach is similar to the "Geometry Rendering" approach to remote rendering described by Luke and Hansen [2002]. The client is still expected to handle the rasterization of all rendered geometry, but that geometry is only a spatial and temporal subset of what the server has available.

7.3.4 Remote Rendering Architecture

Generalized Client-Server Approach

The remote rendering architecture is divided up for a server Unity application and a client Unity application. In each application, a simple Unity Prefab object is placed (Figure 7.3 green-red) and configured for communication. The server application is

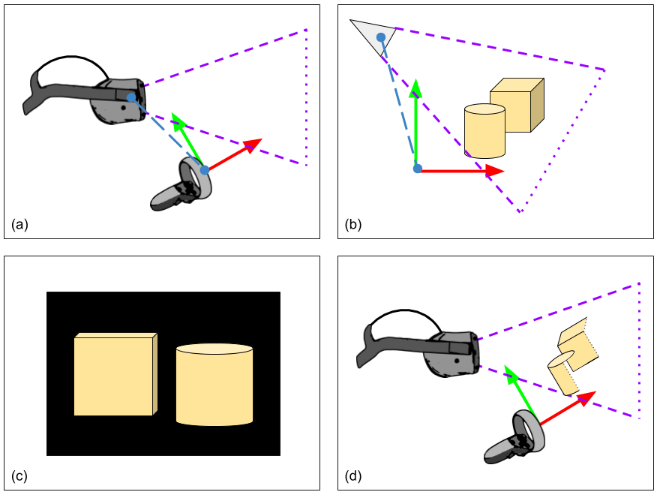


Figure 7.3: a) The client viewer application contains a Unity Prefab (green & red arrows) that serves as an origin for the remoted rendered content, which can be moved dynamically by the user. The user's viewpoint relative to this Prefab is captured (blue) along with camera parameters (purple) and these are sent as a content frame request to the remote rendering server. b) A scene of Unity objects (yellow) is captured by a virtual camera (gray). This camera has camera parameters (purple) and offset (blue) relative to a Unity Prefab (green & red arrows) based on the request received from the client (a). c) The content is rendered as a buffer of pixels, plus any additional structural information required such as per-pixel depth, and returned to the client as a content frame response. d) The received content frame is used to produce a 3D "facade" of the original content, displaying only the features of the content visible from the requested viewpoint. This facade is rendered interactively relative to the client Prefab until a new content response is received.

assumed to contain a set of dynamic, opaque game objects depicting an Artifact-Based Rendering Visualization of a dataset(Figure 7.3b yellow). The client application is assumed to contain head-tracked cameras coming from some type of Unity XR camera rig (Figure 7.3a). This might be driven by packages from Oculus or Magic Leap, or even generic Unity XR cameras.

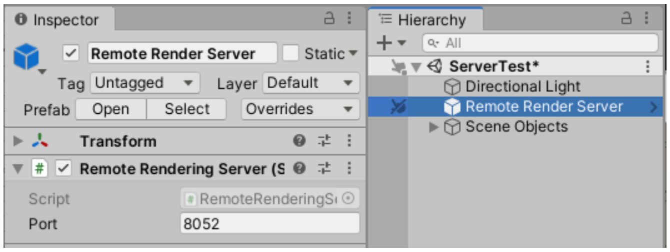
When the server application is launched, it begins listening for any new client connections on a developer-specified port. As each client application launches, a connection is established with the server and a list of tracked views is established for each client. As part of this connection establishment, the server instantiates a virtual camera per-view-per-client.

After connection has been established, the client begins transmitting content frame requests. A content frame is a rendered image of the server's content from a particular viewpoint. In order to request a content frame, the client must provide both a view transform, and all the camera parameters such as resolution, field-of-view, and near/far plane distances. These parameters are generally derived from the head-tracked AR or VR cameras, but can be manipulated for a number of reasons, such as over/under sampling (depending on network constraints), expanded FOV for padding the peripheral of the user's vision, or cropping out background content with a restricted far plane. The view parameters are transmitted to the server in the form of a content frame request.

When a content frame request is received by the server, the server adjusts the position, rotation, and camera parameters of the respective virtual camera (Figure 7.3b gray) in the remote rendering server application to match the tracked client view. The updated virtual camera is then instructed to immediately render the scene to a render buffer (Figure 7.3c), capturing both the rendered RGB values as well as any other useful geometric information such as the depth buffer. The contents of these buffers are optionally compressed, and transmitted back to the client with a header containing the details of the request. The header and buffers are referred to as the content frame response.

When the client receives the content frame response, it first unpacks the buffers into graphics objects such as textures. The client then begins generating a content frame facade in a thread separate from the rendering thread. A content frame facade could, for example, be a displacement-mapped mesh of vertices deformed to match the depth of the captured content as a relief pattern from the requested viewpoint. Once the content frame facade has been constructed, the facade Unity objects are placed relative to the content frame request's camera orientation (Figure 7.3d). Until a replacement content frame arrives, the content frame facades are rendered in real-time relative to the Remote Rendering Client prefab transform (Figure 7.3d green-red) from the instantaneous view of the user.

The Remote Rendering Server prefab transform (Figure 7.3b green-red) can be thought of as the origin of a capture volume, and the Remote Rendering Client prefab transform (Figure 7.3a green-red) can be thought of as a projector of the captured content. The Remote Rendering Client prefab can thus be moved, rotated, and even scaled interactively by the user, or treated as a sub-component of a larger visualization system such as Bento BoxChapter 3. The content frame facade will move around interactively even between new content frame updates, which are always requested in the coordinate space of the Remote Rendering client prefab.



☑ Remote Render Server

Tag Untagged

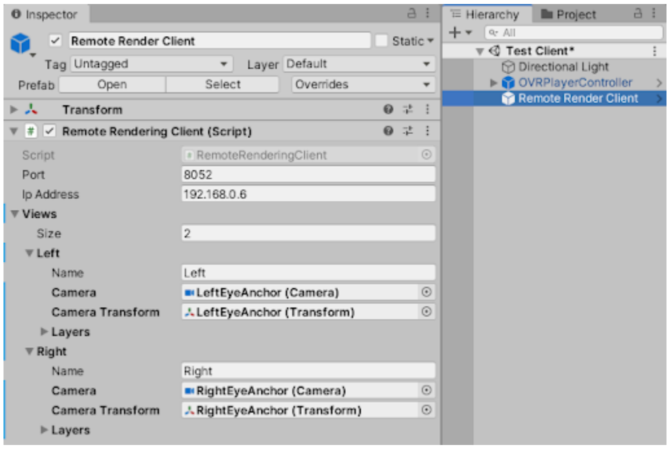
Layer Default

Remote Rendering Server (S

Script RemoteRenderingS

Port 8052

(a) Server-side prefab settings



☑ Remote Render Client

Tag Untagged

Layer Default

☑ Remote Rendering Client (Script)

Script RemoteRenderingClient

Port 8052

Ip Address 192.168.0.6

Views

Size 2

Left

Name Left

Camera LeftEyeAnchor (Camera)

Camera Transform LeftEyeAnchor (Transform)

Layers

Right

Name Right

Camera RightEyeAnchor (Camera)

Camera Transform RightEyeAnchor (Transform)

Layers

(b) Client-side prefab settings

Figure 7.4: (a) The settings required of a developer to add a Remote Render Server prefab object to any Unity Scene. The developer must only specify a port on which the server will listen for new Client connections. (b) The settings required of a developer to add a Remote Render Client prefab object to an AR/VR viewer application. The developer must specify a port, IP Address, and a list of dynamic viewpoints from which to render. In this figure, the left and right eyes of a VR Player Rig have been selected and labeled respectively.

The Unity-based architecture is designed to enable custom remote rendering servers and clients with minimal configuration. The Unity API is provided as a set of Unity Prefabs (short for "prefabricated gameobjects") that can be imported into any existing Unity project.

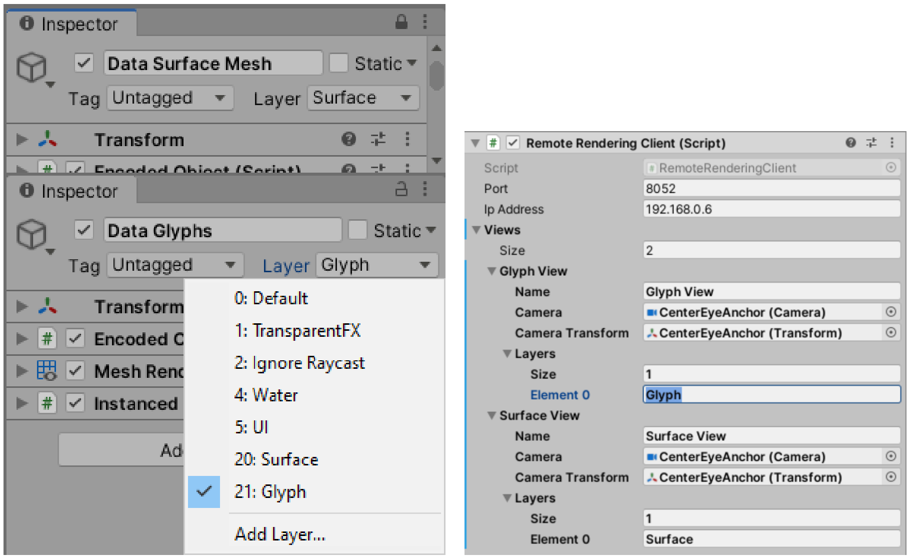
To configure a Unity scene to be a remote rendering server, the developer drag-and-drops the Remote Rendering Server Prefab into the scene hierarchy, and adjusts the networking port if necessary (Figure 7.4(a)). The developer may also choose to position the transformation component of the Remote Rendering Server prefab to center the capture space on the important part of the scene.

To configure a Unity scene to be a remote rendering client viewer, the developer drag-and-drops the Remote Rendering Client Prefab into the scene hierarchy, sets the appropriate port and IP address of the remote rendering server, and specifies how many views should be used for capturing content (Figure 7.4(b)). Each of these views are labeled, and both a camera and a transformation object are selected to provide parameters for the server-side virtual cameras.

Additional optional features may also be configured. For example, a current extension allows the developer to assign different types of objects on the server-side to be exclusively rendered into a particular view's content frame. To use this feature, the remote rendering server application assigns custom layer labels to particular objects of interest. Custom layer labels could be "Glyph" or "Surface" (Figure 7.5(a)). Such layers can be assigned by hand in the Unity UI, or procedurally through scripting. Then, on the client side, the developer can explicitly enter any number of layer labels for each view. For example, two content frame facades can be generated for two different types of geometry if the developer sees fit (Figure 7.5(b)). By not selecting any layers for a client-side view, all layers will be captured.

All of these settings – and more – can be accessed through the script for the Remote Rendering Client script. Besides these configuration settings listed in the Unity UI, additional overrides can be set for the client views, such as field-of-view overrides or resolution multipliers. These can be set dynamically at run-time, allowing the application to optimize for better performance or higher resolution.

These scripts can be found in the example applications found at
`https://github.umn.edu/joh08230/QuestDepthViewer` and
`https://github.umn.edu/joh08230/QuestDepthServer`.
Note: These will be moved to a public-facing repository with documentation.



Data Surface Mesh
Tag Untagged
Layer Surface

Data Glyphs
Tag Untagged
Layer Glyph

Script RemoteRenderingClient
Port 8052
Ip Address 192.168.0.6
Views
Size 2
Glyph View
Name Glyph View
Camera CenterEyeAnchor (Camera)
Camera Transform CenterEyeAnchor (Transform)
Layers
Size 1
Element 0 Glyph
Surface View
Name Surface View
Camera CenterEyeAnchor (Camera)
Camera Transform CenterEyeAnchor (Transform)
Layers
Size 1
Element 0 Surface

(a) Server-side layer specification (b) Client-side layer specification

Figure 7.5: Specifying layers to be captured for each view in the client viewer application

7.3.5 Content Frame Facade Generation

At its simplest reduction, a content frame facade could be a quad placed at the far-plane displaying the remotely-rendered pixels (referred to as Image Rendering by Luke and Hansen [2002]). They describe three rendering approaches that are all integrated into their approach: Image Rendering,. If new content frames could be fetched instantaneously per render frame, this would result in a seamless remote rendering experience. However, network latency and server rendering time will inevitably introduce delay between a user's motion and the resulting change to the content presented to the user's eye. In VR, this motion-to-photon delay is known to cause cybersickness in some users [LaViola Jr 2000].

Instead of relying entirely on the server for rendering, some rendering can be done per-frame on the client device to correct for the user's instantaneous view orientation, driven by the content frame. Some approaches could be based on Geometry Rendering, like rendering imposters [Schaufler and Stürzlinger 1996; Sillion et al. 1997] at coordinates provided by the content frame. Other approaches could be based on ZTex/Depth

Image Based Rendering (DIBR) by using the depth buffer to produce a 3D relief of the content [Heckbert 1986; Oliveira et al. 2000; Tatarchuk 2005]. In both cases, entrusting some limited rendering to the client, the user can make comfortable changes to their point-of-view, receiving per-frame imagery spatially consistent with their motion. Any updates to the content frame will be inserted once they've been received. Because viewpoint correction can be performed on the client side, new content frames need not be acquired for every render frame. In practice, 3-to-10 content frames per second may prove sufficient for a comfortable viewing experience.

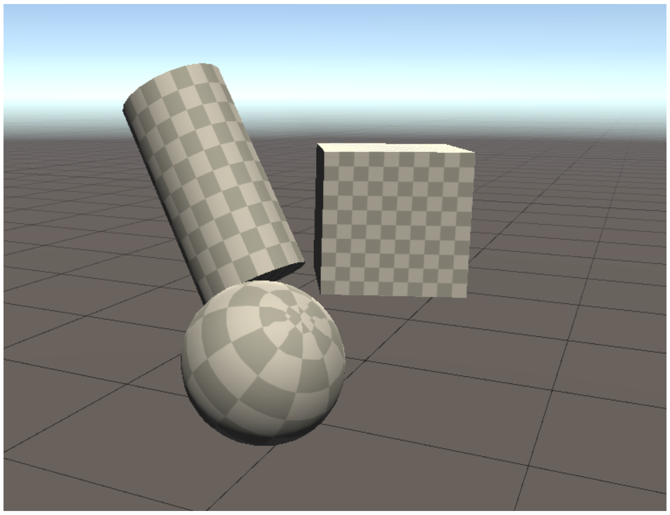


Figure 7.6: A Unity scene of objects on the Remote Rendering Server application

In our implementation, we utilized DIBR to correct for per-frame changes in view. Suppose we have a scene of objects such as those in Figure 7.6. When a new content frame request arrives at the server, the DIBR technique dictates that the scene is rendered to both a colorbuffer and a depth buffer, which are combined into an RGBA texture where the A component is the depth of each pixel normalized to eye-to-farplane space (depth of 0 corresponds to a pixel at the eye, and depth of 1.0 corresponds to a pixel located on the far plane). This depth-encoded image will be called an RGBD texture. The RGBD texture is packaged as a byte array into the content frame response along with the header describing the camera orientation, field-of-view, near/far plane, and resolution of the RGBD texture byte array.

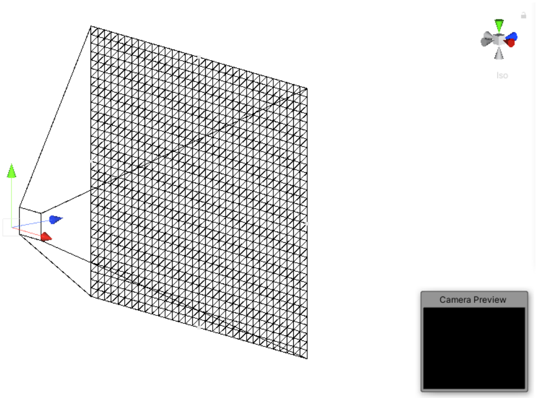
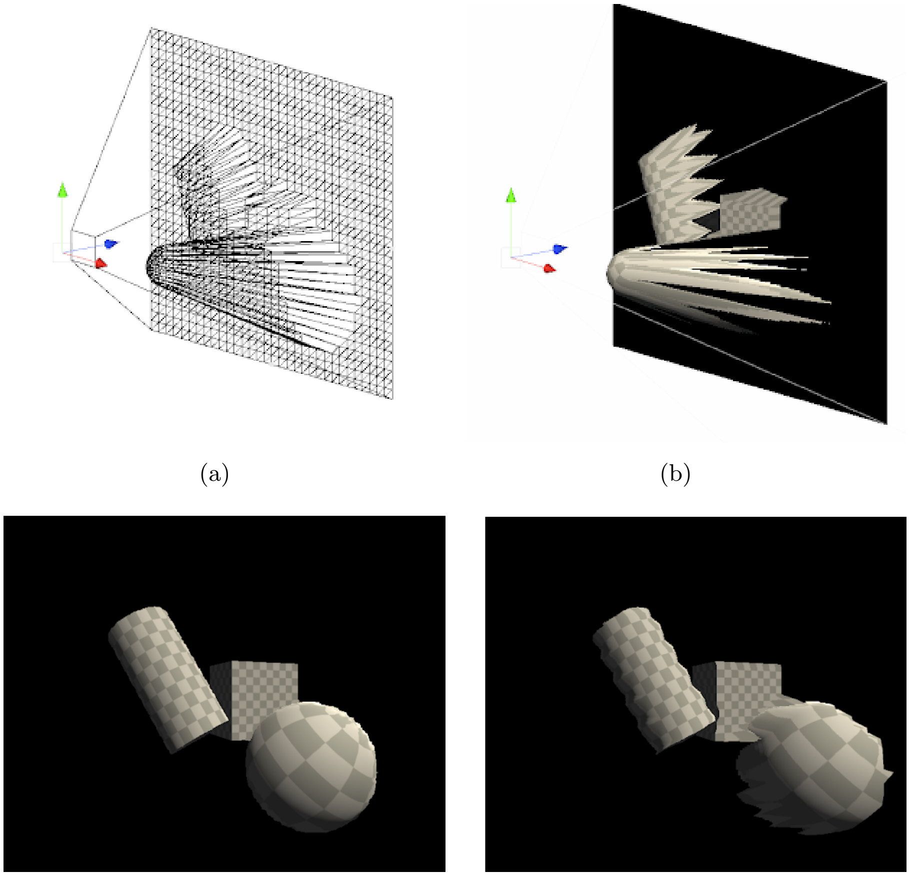


Figure 7.7: A grid of quads aligned at the far-plane specified by the configuration in the content frame response. Here the resolution of the grid has been specified as 32x32 for the purpose of illustration, but in practice grid may be more densely subdivided.

Depth Reprojection

When the DIBR content frame response arrives to the client, the RGBD texture is re-assembled, and a new gridded mesh is aligned at the far-plane of the camera configuration specified by the header of the content frame response Figure 7.7. This grid will have a vertex resolution less than or equal to the resolution of the RGBD texture. In practice, the resolution may be around 1600x1600 pixels in the case of the Oculus Quest, but a 2,560,000 vertex mesh will likely not render at interactive rates on the Oculus Quest. Thus, the developer will likely specify a mesh resolution lower than the resolution of the RGBD image with acceptable results. (The appropriate granularity of the mesh may depend on the distribution and size of the items in the scene.) Until the grid is deformed, it is not displayed to the viewer.

This mesh is then deformed on a new CPU thread. First, the implicit UV coordinate of each vertex is used to sample into the depth component of the RGBD texture.With no additional configuration, these depths are then used to displace each vertex towards the viewpoint used to request the current content frame. By linearly interpolating from the viewpoint position to the vertex's initial position by the sampled depth, the vertex will be aligned coincidentally with the surface of the captured object in the server scene. The resulting mesh appears in Figure 7.8(a), and with the RGB data projected onto it,



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

Figure 7.8: a) The mesh is deformed by displacing each vertex towards the content frame's viewpoint according to the sampled depth. b) The RGB components of the texture are projected onto the mesh from the content frame viewpoint. c) The rendered result from the requested viewpoint. d) The rendered result from a different viewpoint offset towards the left.

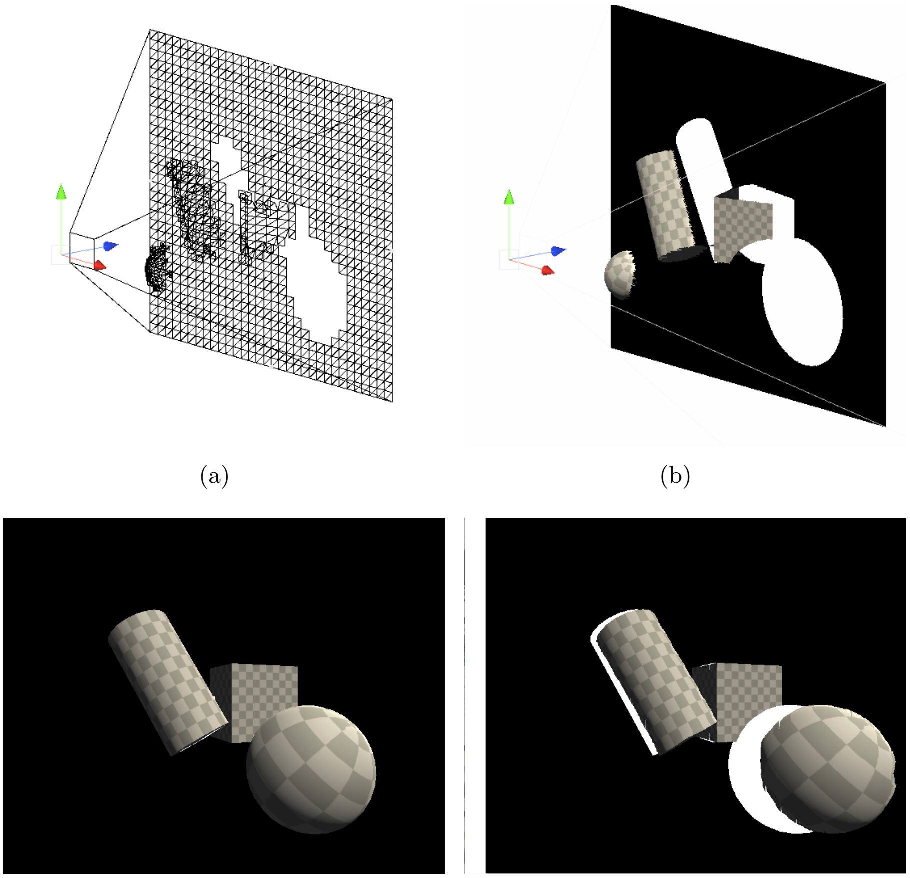
the mesh is textured as in Figure 7.8(b). If the user has not changed their viewpoint since the content frame was requested, they will see the image in Figure 7.8(c). However, if the user has changed their viewpoint in the meantime, they will see something similar to the image in Figure 7.8(d).

Artifact Reduction

Note the smearing of the image around the boundaries of each object in Figures 7.10b&d. These artifacts appear because the mesh is "connecting the dots" between vertices on one object and vertices on another (or vertices corresponding to the far-plane), and can be spatially misleading as the smeared fragments appear to represent a surface that does not exist. While some sort of artifacts are to be expected near the objects' boundaries due to occlusion from the sampled viewpoint [Carmack 2013], steps can be taken to reduce the impact of these artifacts.

Our solution to the smearing problem is to determine which quads overlap the seams between objects of disparate depths, and duplicate these quads for each of the depths between which the quad would otherwise stretch. To determine whether the quad overlaps a seam, the developer specifies a depth discontinuity threshold. After each vertex has been sampled for its depth from the RGBD texture, the depths of each quad are compared to see if they contain a depth discontinuity. This is done by sorting the four corner vertices of the quad by depth, and iterating through them. When a jump in depth exceeds the depth discontinuity threshold, the list of corner vertices is split and the quad is duplicated. Each of the corners corresponding to one of the quad duplicates are considered "true" and are kept, and the remaining corners are guessed. In the current implementation, the corners are guessed by averaging the true corners depths (This could be improved by projecting the guessed corners to a plane produced by sampling depth near the true corners). When the RGB texture is re-projected onto this new mesh, the depth component is also sampled and compared against the depth of the mesh onto which it is being projected. If the difference in depths exceeds the depth discontinuity threshold, the fragment is discarded.

Figure 7.9 shows the result of applying our fix to the smearing problem. Note that smearing in Figures 7.8 (b)&(d) are replaced by an empty space in Figures 7.9 (b)&(d). This space actually occupies a smaller amount of the final image than the smearing did, as colored fragments that existed along the smear now have depth-aligned quads upon which they can be rendered. The space still corresponds to regions occluded in the original view.



(a) (b)

(c) (d)

Figure 7.9: a) Before displacing by depth, the quads of the mesh are duplicated when overlapping a depth discontinuity over a specified threshold. b) The RGB components of the texture are projected onto the mesh, sampling the depth component to determine whether the sample pixel belongs at the corresponding mesh position, or discarded. c) The rendered result from the requested viewpoint. d) The rendered result from a viewpoint slightly off-set to the left.

Background Removal

Often in data visualization scenes, the objects being rendered take up only part of the view plane, leaving much of the screen as unrendered background pixels. Such regions of the RGBD texture are assigned depths of 1. Having split the quads up by depth discontinuity, we can now easily remove the background vertices before submitting the

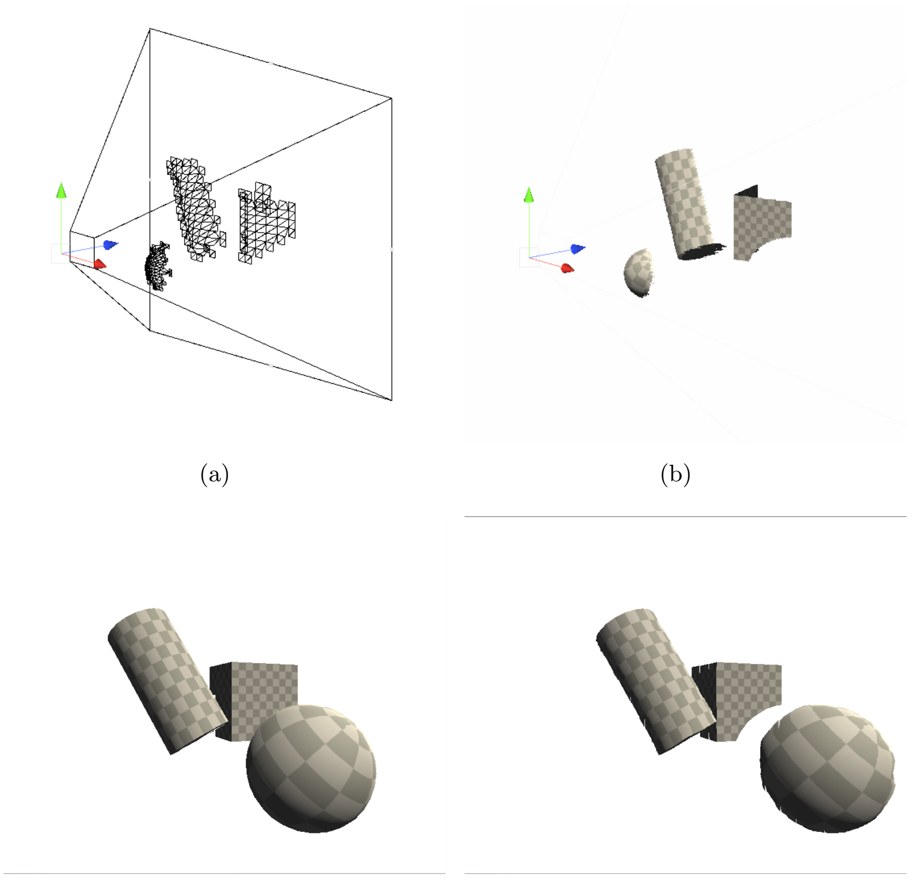
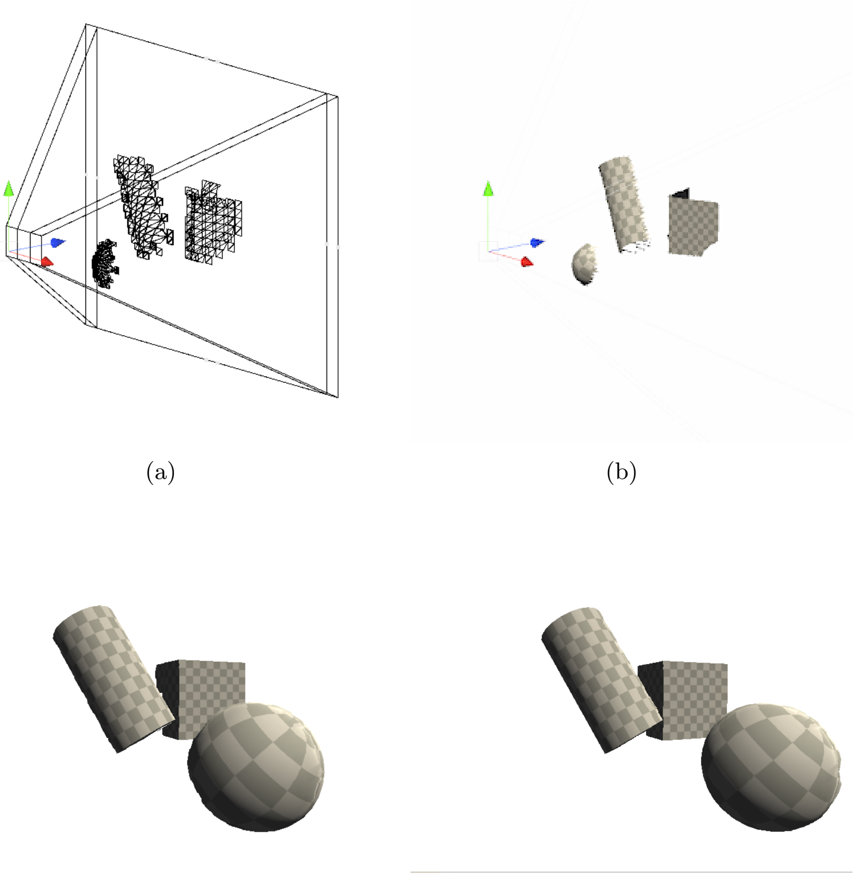

(a) (b) (c) (d)

Figure 7.10: a) The results of smear reduction with quads on the far-plane removed. b) The RGB image projected only on the remaining mesh. c) The rendered result from the requested viewpoint. d) The rendered result from a viewpoint slightly off-set to the left.

mesh to the GPU, saving rendering resources (Figure 7.10).

Multiple Views

If the results of a single viewpoint are able to be rendered with rendering resources to spare, then many of the silhouetting artifacts may be mitigated by superimposing more content frames from offset viewpoints. Responsible candidates for multiple views could be the viewpoints of the left and right eyes, or perhaps extrapolated projections for the



(a) (b)

(c) (d)

Figure 7.11: a) Two meshes placed at the far-planes of two offset viewpoints. b) The two RGB images projected onto their respective meshes and rendered coincidentally. c) The rendered result from the right-most viewpoint. d) The rendered result from near the left-most viewpoint.

possible views. Even if none of the rendered viewpoints precisely correspond with the user's instantaneous viewpoint, the coverage of these multiple content frames will likely fill many of the silhouette spaces (Figure 7.11). Another option is to assign different views to foreground and background items by means of the Layer option (Figure 7.5).

Currently, several configuration options must be heuristically selected by the developer depending on the contents of the scene, such as the granularity of the facade mesh or the tolerance for the depth discontinuity threshold. Future work may find automated procedures for selecting these parameters dynamically based on viewpoint and contents of the scene.

7.4 Performance Characterization

There are several metrics that can be used to characterize the performance of the techniques described in this chapter and compare them against alternatives. Quantitative metrics include such factors as delay time in the fetching of new data geometry and remote rendering content frames, or screen-space error between a content frame facade and a ground-truth rendering of the content. Qualitative metrics concern factors such as user feedback on both the comfort of a remote rendering experience or ability to make accurate judgements on the spatial structure of a 3D dataset.

In this section we provide a set of characterizations that can be used for describing the performance of the methods found in this chapter, and designing future evaluations. With each characterization is given an example technique that could be considered for optimizing that characterization.

7.4.1 Data Streaming Performance Characterization

Since the information being transmitted in our Paraview-to-Unity Data Streaming technique is assumed to be a lossless communication of bytes describing geometric meshes, we are primarily concerned with matters of time.

Data Server Sampling Delay (Time)

Changes in timestep or sampling parameters in Paraview require computation or disk-reading of data, and will have some temporal cost. This cost can be measured as the delta between the time a change is made in the Paraview state, and the results are pushed into the socket connecting Paraview to Unity.

While this characterization is bound to the complexity of Paraview's own processes, large delays could be mitigated by contextually predicting changes the user may wish to make to the Paraview state and pre-compute multiple sets of geometry for immediate response.

Data Geometry Transmission Delay (Time)

Depending on the size of the data geometry (approximately equal to the complexity of the mesh times the number of variables) and the network latency, there will be some delay in the transmission of the data to the visualization system. This delay can be measured as the delta between the time data geometry is submitted to the socket, and the time that it has been fully loaded in the visualization system.

If there is substantial delay from large data geometry, the data stream could be compressed before sending, or the pipeline could be extended to send and receive the data in pieces, enabling the visualization system to begin handling subsets of the data geometry asynchronously as the socket is read.

7.4.2 Remote Rendering Performance Characterization

The primary motivation for our approach to remote rendering over existing techniques like ALVR is to prioritize responsiveness to head tracking, low motion-to-photon latency, and provide a comfortable, non-cybersickness-inducing experience for wider audiences. To this end, a key set of our remote rendering characterizations relate to user comfort and the delay from the capturing of head-tracking data to the display of view-specific imagery.

Admittedly, our technique avoids cybersickness by introducing a tradeoff in the fidelity of the imagery. While we accept the sacrifice of a flawless reproduction of a locally-rendered experience, it is still critical that remote rendering a visualization still provides useful utility for exploring the data. Thus, a secondary set of characterizations relate to user feedback on the usefulness of our remote-rendered visualizations, as well as factors such as the pixel-per-pixel reproduction accuracy and delay in presenting new perspectives of the data as the user explores the data space.

In this section we will provide a description of both the quantitative and qualitative metrics that characterize both the issues of cybersickness and analytic fidelity.

Cybersickness, Quantitative Characterization

Viewer Instantaneous Framerate (Frames per second)

This characterization is the actual frames-per-second of the AR/VR client viewer app. This framerate should be targeted to the max refresh rate of the client viewer device (e.g. 72Hz for the Oculus Quest) in order to avoid cybersickness. This measure primarily describes the rate at which all the visible content frame facades can be drawn based on instantaneous head-tracking data. The most direct factor influencing the Viewer Instantaneous Framerate is the number of vertices in the combined set of active content views.

In our implementation, the easiest way to improve the framerate is lowering the grid dimensions for the content frame facades. Additional optimizations could likely be developed for dynamically simplifying the facade meshes, such as content-aware decimation. We've already taken some steps in this direction by employing the removal of background quads (Figure 7.10).

Cybersickness, Qualitative Characterization

User feedback (User evaluation on comfort)

One of the reasons we are concerned with the previous Viewer Instantaneous Framerate metric is to avoid cybersickness. However, we cannot overlook the importance of user feedback on the experience. There may be unforeseen effects of certain remote rendering techniques on user comfort beyond framerates, and so gathering some measure of users' experience may help in refining the chosen techniques and parameterizations.

There are several methods for determining degrees of cybersickness, such as administering questionnaires, tracking postural sway, and evaluating physiological state [Rebenitsch and Owen 2016]. As we are primarily interested in the amount of cybersickness introduced through our remote rendering methods, and cybersickness-related experiences vary between people, it is important to compare a user's experience with remote rendering of a particular scene on a specific affordable untethered AR/VR device against experiencing the same scene rendered locally on the same head mounted display.

One possible example of another contributing factor to cybersickness could be situations where the remote-rendered scene contains many glyphs or other features that are small enough to sometimes fall between the vertices of the content frame facade. In such circumstances, some subset of glyphs may not appear during occasional content frames. This would result in a "popping" in-and-out of some foreground elements, which may have an undesirable effect on the user. If this is found to be a contributing factor, this issue can be mitigated by increasing the resolution of the content frame facade.

Analytic Fidelity, Quantitative Characterization

Content Framerate (Content frames per second)

This characterization describes the rate at which new content frames are displayed to the user. The higher this measure, the more quickly negative artifacts from changes in viewpoint will be reduced, but also the faster changes to the remote visualization will be made visible to the user. The developer can chose to request new content frames as frequently as they deem appropriate, keeping in mind that if the content request rate exceeds the reciprocal of the Content Frame Fetch Delay, multiple content frame facades may need to be generated simultaneously, which may tax the computational resources of the device.

For our implementation, we've capped the content framerate per-view to the reciprocal of the Content Frame Fetch Delay by waiting to send a content frame request until the last content frame fetch has been completed. This was to avoid back-logging content frame responses. However, the Content Framerate could be increased depending on network latency by intelligently queuing requests and responses based on monitoring the expected sub-characterizations of the Content Frame Fetch Delay. This way, content frame responses could be timed to arrive approximately when the last content frame facade has finished generating.

Content Frame Fetch Delay (Time)

The entire round-trip time from the moment of making a content frame request to the moment that content frame's facade is made visible to the user is dictated by a number of sequential sub-characterizations.

- **Content Frame Request transmission Delay:** This is the measure of the time it takes to collect the viewpoint parameters on the client and transmit them via a socket to the server. As the Content Frame Request is a simple struct on the order of 100 bytes, this sub-characterization is entirely dependent on the network latency.

- **Content Frame Rendering Time:** This is the measure of the time it takes to render a single frame of the server-side visualization scene from a content frame request viewpoint. Since the client does not need a unique content frame per second, this rendering time does not even need to maintain a VR-level framerate (60-90 frames per second, or 0.17 to 0.11 seconds per frame) for remote-rendering purposes. However, if many clients are connected, these rendering times may begin to add up.

- **Content Frame Response transmission Delay:** Like the first sub-characterization, this measure is largely dependent on network latency, but can also be impacted by the size of the content frame response RGBD data. This can be mitigated by both reducing the resolution of the content frame, and compressing the RGBD data. Both of these come with tradeoffs, however: lowering the resolution will directly impact the quality of the viewer's experience, and compression will require decompression on the client viewer, which may run on hardware with greater overheads for handling decompression.

- **Content Frame Response Unpacking Delay:** This accounts for any decompression and re-assembling of the RGBD data into a texture that can be sampled. This may be largely impacted by hardware capabilities, as well as visual complexity of the RGBD data when compression has been used.

- **Content Frame Facade Generation Time:** The factors contributing to time required to generate a content frame facade depend primarily on implementation. In our implementation, this time is approximately linear to the number of grid vertices in our DIBR mesh, but has a rather high linear coefficient from the determination of quad duplication.

Viewer Resolution (Content Frame pixels per Screen pixels)

This is a measure of how much pixel density is lost from the original visualization system, primarily due to altering the content frame resolution multiplier. By default, the content frame resolution multiplier is 1, which means that the content frame provides on RGBD pixel per viewer screen pixel. Assuming the user does not move their view significantly towards the facade before a new facade is produced, the RGB resolution of the facade will remain approximately equal to the viewer's screen resolution. However, if the resolution multiplier has been reduced in order to improve the Content Frame Fetch

Delay (and subsequently the Content Framerate), the resolution of the facade will suffer. It is up to the developer to determine contextually whether the pixel resolution of the facade is worth sacrificing the rate of new content frame acquisitions.

It is worth noting that this only relates to content frame pixels per screen pixel, and if the user moves their viewpoint closer to the remotely-rendered content, new facades will display the same region of the visualization more clearly than the same region viewed from afar.

Viewer Missing Pixels (Difference in Instantaneous Viewer Pixels vs Instantaneous Ground Truth Pixels)

This is a quantitative measure of how much visual information is lost from remote rendering as the user's viewpoint changes. In our implementation, this is most acutely related to the cut-out silhouette regions (For example, the white regions of Figure 7.11d). To measure this characterization, a test could be conducted rendering a set of geometry on both the server and the client, and capturing both the remote rendered content frame facade and the locally-rendered geometry, and calculating a difference heuristic for the two frames.

This measure should be expressed as a function of total displacement between the instantaneous viewpoint and the requested viewpoint for the current content frame. The result will of course vary according to the visualized data and the nature of the view position. According to Carmack, "A scene with no silhouette edges, like the inside of a box, can be warped significant amounts and display only changes in texture density, but translation warping realistic scenes will result in smears or gaps along edges" [2013]. Thus, calculating average and worst missing pixel metrics for a variety of scene types would be useful for guiding a developer to target appropriate content frame rates for their type of visualization.

As shown in Figure 7.11, one good way to mitigate missing pixels is to add additional offset views. Also, the mixing pixel metric can be expected to increase as the current viewpoints deviate from the requested viewpoints, and so choosing settings to increase the content frame rate can also help.

Analytic Fidelity, Qualitative Characterization

Viewer Negative Visual Artifacts - Analytical Accuracy (User evaluation on observations of data structure)

It is critical that our techniques for remote rendering 3D data visualizations provide utility for actually viewing 3D data visualizations. A key motivation for pursuing head-tracked, stereo remote rendering of visualizations is based on Aygar, Ware, and Rogers' assessment that these features are important for understanding the 3D structure of many datasets [2018]. Thus, we want to be mindful that we're maintaining that analytic utility as we transform the data into a remotely-rendered facade.

Aygar et al. describe experiments that were used for determining the impact of stereoscopic and motion-based techniques on users' perception of the spatial structure of 3D datasets. Such experiments could be adapted for comparing how our Remote Rendering techniques impact 3D perception as opposed to locally-rendered visualizations, or even other remote rendering approaches like 360 video streams. Another approach could be getting expert user feedback from users who have a strong familiarity with the data being visualized to determine how much analytic fidelity is being lost through remote rendering.

7.5 Results

We've reached several critical milestones in the development of both the Data Streaming and Remote Rendering components of this research. We've had opportunities to demonstrate both systems to various audiences, and have captured initial results for several of the aforementioned performance characterizations. In this section, we will describe our progress on all these fronts.

7.5.1 Implementation of Data Streaming

Our current Data Streaming implementation consists of two parts: A plugin for Paraview, and C# code for receiving the data stream from Paraview in Unity applications.

The Paraview plugin is available in our research lab's GitHub repository as a utility that can be downloaded in an XML format and loaded into any instance of Paraview

5.6.0. Many of our team members are actively exploring new datasets in Paraview and using our data steaming code locally on their systems for quickly accessing Paraview data objects in our Unity Artifact Based Rendering application. This Paraview plugin can also transmit data objects to any application that can read VTK datasets, such as a python script. Such an application need only open up a listener on a TCP port and wait for a connection on that port from Paraview. The Plugin will transmit a label string for the incoming data object, followed by an ASCII encoding of the VTK data. In this way, the plugin can serve many Data Streaming workflows, not limited to Unity-based systems.

Currently our C# implementation of the receiving end of the Data Streaming technique is integrated into our Unity ABR Engine GitHub repository. The `ParaviewDataListener` script receives the labeled VTK data objects and converts them into ABR-ready data meshes.

7.5.2 Demonstration of Data Streaming

In September of 2019, the National Science Foundation held an unveiling event for a new supercomputer at the Texas Advanced Computing Center (Figure 7.12). Our team of collaborators from Chapter 5 worked together in Austin to prepare a demonstration of multiple timesteps of bio-geochemistry data from the Gulf of Mexico on the supercomputer being sampled to streamlines and glyphs, which were sent in real-time to a laptop attached to an HTC Vive Pro which we used to show visitors an ABR visualization designed by our artist.

Thanks to our Data Streaming technique, we were able to present to the public real-world examples of supercomputer-scale datasets in an interactive virtual reality visualization.

7.5.3 Performance & Discussion of Data Streaming

Paraview was running on a single node of Frontera, loading in a timestep data from disk, doing the visualization and passing the data to Unity. In our use case, we were only interested in a relatively small subset of the data - the gulf, which is a small fraction of the overall, world-wide simulation results. We culled that subset off-line, so

(a) Checking out one of the supercomputers at the Texas Advanced Computing Center, home of Frontera, the supercomputer we used for data streaming

(b) An ABR visualization being shown to many journalists, researchers, and NSF representatives via data streaming from Frontera during an NSF event

Figure 7.12: Two images taken by photographers in Austin during an NSF event unveiling a new supercomputer called Frontera. At this event we presented a demo of live data streaming from Frontera to a laptop running an ABR visualization.

the run-time Paraview task - loading in the subsets and generating the geometry - was well within the capability of a single node, generating new time steps of visualization output to Unity every few seconds. In this particular case, the task being performed didn't necessitate a supercomputer, but rather a strong compute node with immediate access to the data. The advantage here is in not having to ship the all the time steps of the volumetric data out of the computer room and, instead, ship only the sampled geometry required for visualization, which are orders of magnitude smaller.

Our two performance characterizations for Data Streaming, Data Server Sampling Delay and Data Geometry Transmission Delay, are both highly dependent on the complexity of the datasets with which we work. When using small pieces of procedurally-generated test data, both delays can be a matter of milliseconds. However, larger datasets often result in over a second of delay for both sampling and transmission.

During our Supercomputer demonstration, the re-sampling of streamlines and pointsets took approximately 4 seconds of supercomputer time for each new timestep, and the transmission of the resampled streamlines and pointsets took about 0.25 seconds. However, the transmission of the bathymetry of the Gulf of Mexico took about 5 seconds due to its high-resolution surface mesh grid of over a million points, each with normal, height, and longitude/latitude variables. Fortunately, we only needed to transmit the bathymetry once as it did not change with new timesteps.

Both of these metrics are highly dependent on how spatially complex the data is, and how many variables need to be sampled or transmitted. It's worth remembering that the size of a VTK dataset scales linearly with the number of variables present in the dataset, and we have often found it prudent to prune down the variables we aren't presently interested in before configuring the data stream.

In a more complete application, users could immersively explore the entire world-wide dataset by using many supercomputing nodes to handle sampling, and feeding viewport information to the supercomputer so it would prioritize sending the portion of the visualization sample geometry most likely to be seen. Such a system could be end up with a streaming solution directly analogous to the remote rendering approach, where, as the user moves their head, they might see a blank area that would fill in as newly revealed data is received. This would required the building of a distributed application with subprocesses attaching to the subprocesses of Paraview across the distributed system, receiving the data in a many-to-many parallel fashion, then gathering and transmitting prioritized subsets of it based on viewing information being received. Additionally there could be feedback directly to distributed Paraview also to perform user-requested sampling, such as selecting iso values or seeding streamlines.

7.5.4 Implementation of Remote Rendering

The Remote Rendering technique, as described in this paper, has been developed and made available as both an open-source GitHub repository and a Unity Package. By including this code in a Unity project containing a data visualization system, a scene can be quickly configured as a Remote Rendering server. And by adding the code to an Oculus Quest, Magic Leap, or any other interactive scene, a client viewer can be created.

The Remote Rendering source code depends on another open-source codebase available on our research lab's Github, a generalized Unity Utilities library we've been developing in parallel. This provides useful logic such as queuing Unity-specific code from asynchronous or threaded code. (Many Unity methods and classes can only be accessed on the rendering thread.)

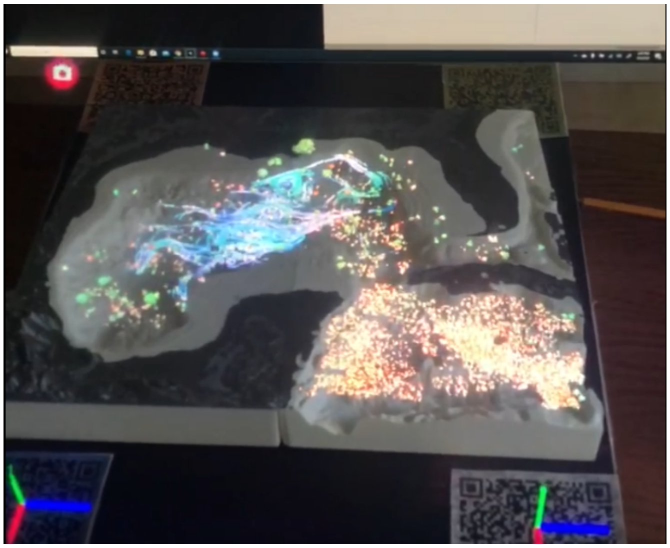


Figure 7.13: An ABR visualization being remotely rendered into a Magic Leap to be viewed with a 3D printed bathymetry.

We've already had multiple team members use this Remote Rendering code to create their own client viewers. The Magic Leap viewer was created by one student in less than an hour (Figure 7.13), and another student quickly produced a 2D desktop remote rendering viewer that lets the user interactively orbit around the remotely rendered content with a click-and-drag of the mouse.

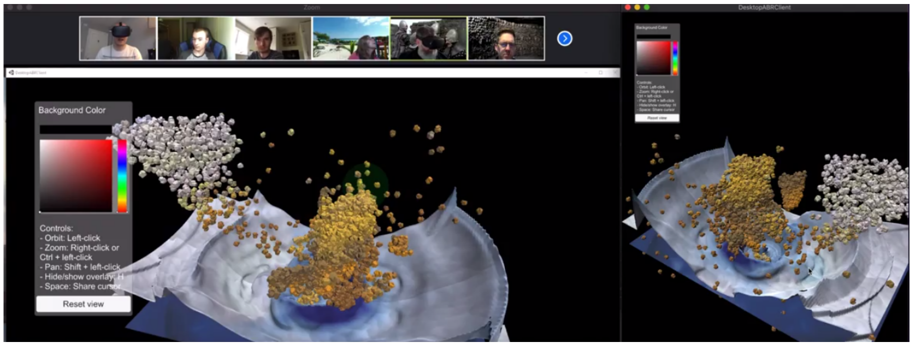
We've also integrated Remote Rendering into our ABR application, so any instance of ABR can serve as a remote rendering server.

7.5.5 Demonstration of Remote Rendering

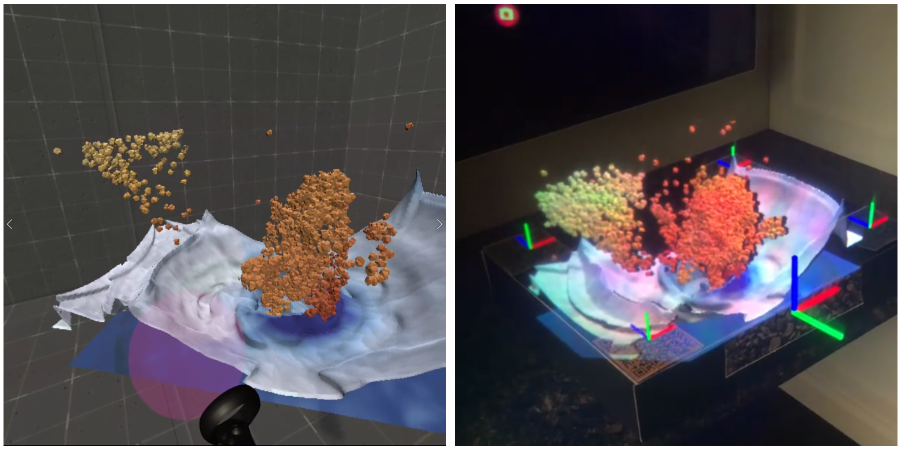
We have demonstrated Remote Rendering in the form of an interactive group design discussion across multiple geographic regions in May of 2020. In this demonstration, we've used Data Streaming to transmit Paraview data hosted in one midwestern city to a computer running ABR in another midwestern city, and had 7 collaborators interactively viewing the same visualization simultaneously in VR, AR, and 2D on Oculus Quests, Magic Leaps, HTC Vives, and Windows & OS X desktops from several US states (Figure 7.14).

Format

This demonstration was performed in the setting of our weekly Artifact-Based rendering team meeting, between regular participants in multiple states (including Minnesota, Texas, and Massachusetts). These meetings often consist of design discussions about how best to apply ABR visualization techniques to new datasets, and previously such discussions have been conducted through screen-sharing using Zoom or Skype. However, through the introduction of our remote rendering support and a new, under-development ability to interact with the ABR design interface remotely, we were able to interact and explore the developing visualization more fluidly than before.



(a) Two Desktop remote rendering clients showing different views (Left: a Zoom screen-share of a Windows client. Right: a Max OSX client.)



(b) A view from an Oculus Quest client. (c) A view from a Magic Leap client.

Figure 7.14: Our remote design discussion about an asteroid impact dataset being viewed on many types of devices concurrently.

In this group design discussion, we considered several possible visualization design options for an asteroid impact dataset across multiple timesteps. The design discussion included our software engineering team, an Artist, a collaborator who had been directly involved in the development of the scientific data, and other participants from multiple disciplines. Our main discussion points revolved around the coloring and sampling of the glyph field encoding the air pressure distribution following the simulated impact. And for comparison, we dynamically switched to other ABR-visualized datasets including the Gulf of Mexico dataset.

Based on real-time feedback, we updated the glyph sampling and data visualization on-the-fly, and participants were able to see the changes in interactive VR, AR, and with 2D desktop views. Participants were also able to explore their own personal view of the data with tracked controllers, image-tracked props, or their computer mouse. We also provided a feature where users could click or hold a button on a tracked controller (in the case of the Vive, Magic Leap, and Oculus Quest) to broadcast a spherical cursor to the other viewer applications as a way to draw attention to different regions of the visualization.

Results

We ended up maintaining at least 6 remote rendering clients for 30 minutes, and at had up 12 simultaneous client connections for testing purposes. Client apps were connected via residential internet connections, and each had anywhere from one to four content frame views (fewer are required for monoscopic desktop clients). We had two Magic Leaps, three Oculus Quests, one HTC Vive connected to a Windows laptop, three OSX clients, and three Windows Desktop clients.

Instantaneous frame-rates of at least 60 fps were reported, except in cases where Quest users adjusted settings for higher resolution quality, which would in some instances drop the framerates below 60, causing noticeable tracking lag. Based on data collected from the server application, content framerates generally varied from 3 content-frames-per-second to 0.2 (one content frame update every 5 seconds), depending on requested image resolution, network latency, and device capabilities relating to unpackaging the content frame response.

Feedback on the general remote rendering experience was largely positive. The ability for each participant to re-orient their view, even just on the desktop applications, was reported to be a major improvement from simply watching a screen-share session. Combined with the remote access to the ABR design UI, our artist was able to quickly jump in and make tweaks to colormaps while explaining to the rest of the team what she was doing while everyone watched from their own perspectives. The HTC Vive user, who had worked on the production of the datasets we were viewing, reported a very smooth and comfortable VR experience, but also reported that some glyphs seemed to disappear occasionally depending on their size (this is a problem that sometimes arises when the size of the facade mesh quads exceed the size of the glyphs). Quest viewers also reported a smooth, comfortable experience with the exception of one bug where clients were being auto-disconnected after 3 seconds of no new requests, causing them to re-connect with occasional momentary tracking errors, causing startling jumps in the remote-rendered content.

By the end of the session, participants looked forward to future collaborative design sessions using a similar process. The artist even spoke of looking forward to future remote workshops or tutorials where artists could work together on visualization design using applications like the ones used in the demonstration. None of our AR / VR users reported motion-sickness or discomfort beyond the aforementioned bug (which has since been fixed).

7.5.6 Performance & Discussion of Remote Rendering

Data Collection

While strategies for collecting metrics for each of the characterizations in Section 7.4 is part of ongoing work, we have been able to collect results for several of the time-specific characterizations under a number of different parameters.

To collect the data in this section, we used an Oculus Quest as the client viewer, with a 1440 × 1600 pixel, 72 hz display per-eye. For networking, the quest was connected to a Verizon LTE mobile hotspot. As a remote rendering server, we used a desktop PC with an Nvidia GTX 1070 graphics card, 16 GB of RAM, and an Intel i7-4790K Processor. This computer was connected via ethernet to a Century Link 100 Mbps internet connection. While both nodes were geographically located in Richfield, MN, the

Quest was requested and receiving content frames through a cellular network, similarly to how future 5G-enabled AR or VR displays may function.

In Figure 7.15, we can see three different values selected for both the RGBD image resolution and the the facade mesh resolution. From left to right, the pixel resolution increases, and from top to bottom the facade mesh resolution increases. The center is the setting closest to that most often selected for our demonstrations. In Table 7.1, You can see nine columns, each displaying the specific resolutions, as well as the mean and standard-deviations for several of the time-related characterisations from Section 7.4 collected across approximately half a minute of data capture. Note that for this collection, we only requested a single view, and only requested an update once the previous content frame had been fully realized as a facade.

Only in one high-facade-resolution case do we ever see viewer instantaneous framerate drop below 70 FPS. When multiple views are being drawn with high facade resolution, viewer instantaneous framerate will often suffer, as the number of rendered vertices can exceed the Quest's recommended workload.

For the metrics measured in milliseconds, note that Content Frame Fetch Delay describes the duration from when tracking information is sent in a content frame request to the moment it first becomes visible to the user. During this duration, a number of different process occur in order, including network communication, rendering on the server, and construction of the facade. In this specific data collection, we report both the entire Content Frame Fetch Delay as well as the server time spent rendering and the content frame facade generation time.

As our focus has been on optimizing the Viewer Instantaneous Framerate, we have not focused on optimizing the Content Framerate. To calculate the actual Content Framerate in our example, you can divide the Content Frame Fetch Delay by 0.001. But keep in mind that multiple content frames can be fetched and processed asynchronously. With multiple views or careful timing of requests, a higher Content Framerate could be achieved. Indeed, the primary bottleneck for Content Framerate will likely be the Content Frame Facade Generation step, as this step is costly for the client device. Perhaps by offloading some of the Content Frame Facade Generation to the server could alleviate the burden on the client device.

top=1.5in, left=1in,right=1.5in,bottom=1in

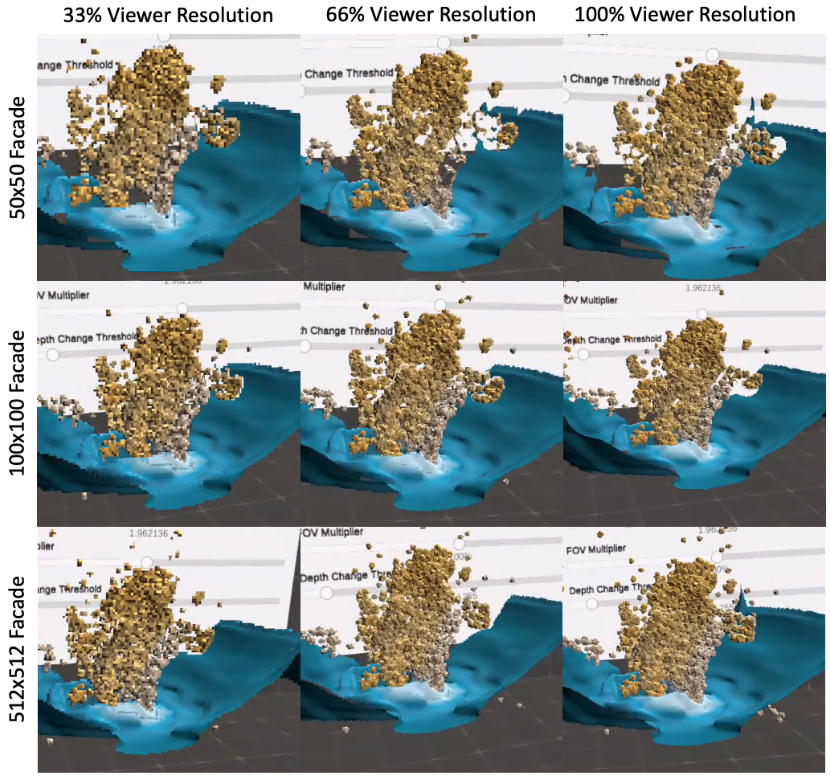


Figure 7.15: The visual results of adjusting two key remote rendering parameters, the pixel resolution of the RGBD content and the vertex-resolution of the facade mesh. For each parameter, three values were selected. Note that when the facade resolution is low, blocky gaps are more likely to appear, and glyphs have a higher chance of disappearing.

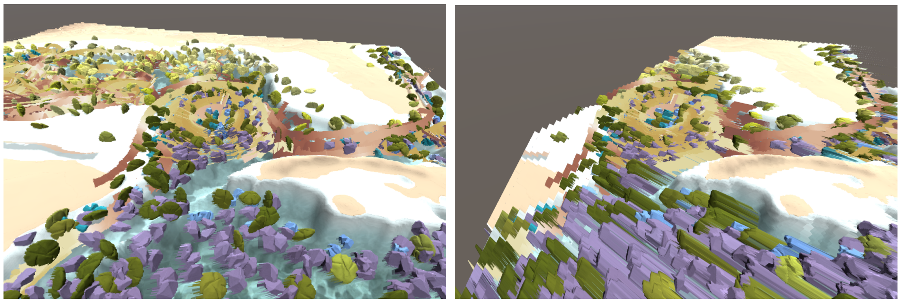
Facade Resolution	50 x 50	50x50	50x50	100x100	100x100	100x100	512x512	512x512	512x512
Viewer Resolution %	33	66	100	33	66	100	33	66	100
Viewer Resolution	450 x 407	887x803	1346x1218	444x402	887x803	1351x1223	444x402	888x803	1352x1223
Samples collected	98	90	77	90	79	78	12	16	22
Viewer Instantaneous Framerate (FPS)	μ:71.45 σ:4.22	μ:71.94 σ:5.4	μ:72.36 σ:2.23	μ:72.33 σ:7.27	μ:73.99 σ:13.44	μ:72.83 σ:3.81	μ:68.76 σ:10.01	μ:71.91 σ:3.2	μ:72.57 σ:3.58
Content Frame Fetch Delay (ms)	μ:350.08 σ:65.07	μ:359.11 σ:14.77	μ:498.39 σ:36.03	μ:450.09 σ:18.93	μ:465.99 σ:27.46	μ:588.74 σ:36.15	μ:3376.17 σ:360.91	μ:3302.56 σ:274.55	μ:3379.95 σ:78.28
Content Frame Rendering Time (ms)	μ:42.63 σ:4.19	μ:45.49 σ:5.33	μ:50.7 σ:5.89	μ:44.58 σ:4.83	μ:46.81 σ:4.99	μ:51.26 σ:7.33	μ:50.58 σ:7.17	μ:51.19 σ:10.91	μ:59.68 σ:5.53
Content Frame Facade Generation Time (ms)	μ:33.17 σ:3.82	μ:31.99 σ:0.76	μ:31.91 σ:0.57	μ:130.6 σ:9.39	μ:126.82 σ:5.36	μ:125.03 σ:6.34	μ:2670.92 σ:285.29	μ:2605.44 σ:213.56	μ:2529.64 σ:14.48

Table 7.1: Results from collecting approximately 30 seconds of data from capturing the parameters in Figure 7.15. For this collection, only a single view was being rendered, and a new content frame was only requested after the previous content frame facade had been presented to the viewer. Note that, as described in Section 7.4, Content Frame Fetch Delay is composed, in part, of Content Frame Rendering Time and Content Frame Facade Generation Time. Also note that we are not tracking Content Framerate, as we are not currently focused on optimizing this metric.

Cybersickness

Our initial motivation for researching beyond existing remote rendering techniques like ALVR was to mitigate cybersickness from motion-to-photon delay. Thus, we represented remote rendering content as 3D geometry and prioritized the instantaneous framerate of the viewer. So did the steps we took accomplish our goals of reducing cybersickness?

We presented development demonstrations of our remote rendering technique at various stages to the same team members who reported cybersickness using ALVR. In all cases, we displayed remotely-rendered imagery of the Gulf of Mexico dataset on the Oculus Quest.



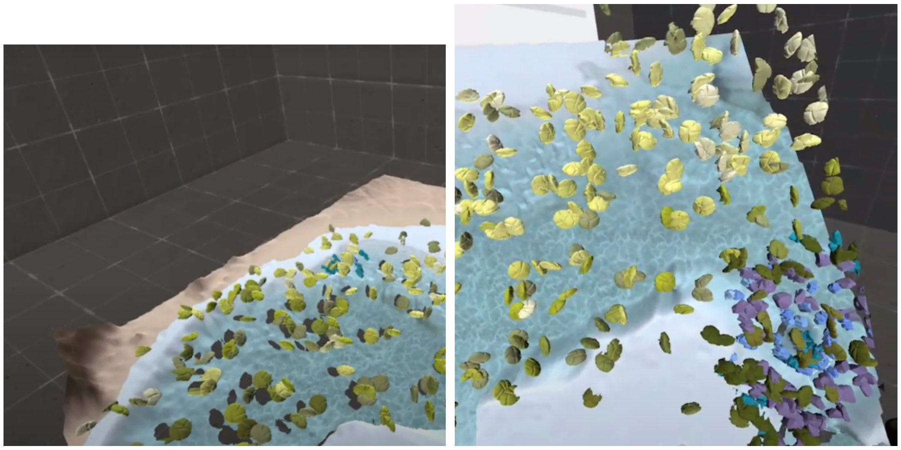
(a)

(b)

Figure 7.16: Varying degrees of distortion from an early stage of the remote rendering project. (a) Small amounts of smearing between glyphs and the background when the user has moved a few centimeters from the requested viewpoint. (b) Large amounts of smearing when the user has moved about 90 degrees around the scene from the requested viewpoint (an extreme example).

Our first development presentation of our new remote rendering showed the content frame facade against a solid background with no artifact reduction (Figure 7.16). User feedback reported that while there didn't seem to be any obvious issues with motion-to-photon delay, the smearing distortion combined with the lack of environmental reference frames in the gray void still resulted in a disorienting experience.

Based on this user feedback, we added several artifact reduction techniques, such as smear removal (Described in Figure 7.9) and multi-layer rendering (Figure 7.5), along with a simple room environment locally rendered in the background of the client viewer (Figure 7.17). When we demonstrated these improvements to our cybersickness-inclined



(a) (b) A view from a Magic Leap client.

Figure 7.17: (a) The result of adding a locally-rendered room environment with smear removal integrated, leaving silhouette holes in occluded geometry rather than 3D smears. (b) Additional artifact reduction by using separate remote rendering views for Surface and Glyph layers in Unity

users, they reported that the experience was "much better" than using ALVR, and did not report any disorientation or nausea.

In all these demonstrations, we have successfully prioritized maximum framerates on the Oculus Quest. Depending on quality settings, we were able to maintain 72 frames per second on the Quest, with only the Quest's own motion-to-photon delay since the geometry visible to the user was being dynamically projected based on the latest head tracking data.

In this way, we met our primary goal of alleviating cybersickness in remote-rendered data visualizations.

Analytic Fidelity

As previously discussed, prioritizing responsiveness to head tracking data has required some trade-offs, such as decreased pixel resolution, artifacts around depth discontinuities, and delay in acquiring new content frames. We've found that these are all interrelated trade-offs, and any of them can be improved at the cost of the others.

For most of our demonstrations, we chose a pixel multiplier of 75%, which resulted in rendering a 1200x1080 pixel image for the Quest's 1600x1440-per-eye display. We chose a mesh resolution of 100x100 quads. And we rendered multiple views to resolve most depth artifacts: for the left eye, we rendered one view of the Glyph layer, and one view of the Surface layer; and for the right eye, we rendered both layers together. We found these parameters to be satisfactory for the visual fidelity of our visualizations, and resulted in a Content Frame Fetch Delay of about 1.5 seconds. By staggering when each of the four views is requested (and subsequently staggering when the new content frames are displayed), we get about three Content Frames per second.

However, the Content Frame Fetch Delay is not consistent, and depends on the content of the scene for two reasons. First, we've discovered that depending on the visual complexity of the frame (e.g. amount of non-background pixels and high-frequency details), compressing and decompressing the RGBD buffer may actually take longer to package, transmit, and unpack the RGBD data than sending uncompressed RGBD data, even though the size can sometimes be 10 times larger. Second, depending on the distribution and size of foreground and background objects, more mesh quads may contain several depth discontinuities, and thus more quads may need to be injected, adding additional computation time into the generation of the Content Frame facade.

In the end, our users mostly did not report the visual fidelity or rate of new content to be detrimental to a useful group collaboration, and found remote rendering to provide a much more favorable collaborative experience over simply viewing a 2D, non-interactive screen share. One participant pointed out that sharing 2D content over Zoom or Skype also can have analogous artifacts like compression artifacts or inconsistent frame rates. While participants agreed that they would prefer a locally rendered experience, the ability to share a remote a multi-user experience including devices like the Quest and Magic Leap is a valuable tool.

There is certainly room for improvement, for example the reports of occasionally missing glyphs in the group design session. This problem can be mitigated by increasing the vertex density of facade mesh, at the cost of instantaneous framerate. Future work is expected to produce more efficient and accurate facade meshes.

While we believe that there is a lot of potential to improve the visual fidelity of our technique, we have concluded that we have preserved a useful degree of utility in spite of the trade-offs introduced by our cybersickness prevention approach.

7.6 Roadmap

While not in the scope of this chapter, there are a number of promising next steps for generalizing and improving remote rendering, and better understanding how different factors impact user experience and performance.

7.6.1 Formal Evaluation of Performance Characterizations

As work progresses in Remote Rendering, it will be important to develop an approach to evaluating the successes and uncover unexpected regression. This will be important to do both quantitatively (e.g. measuring the changes in framerates, pixel differences, etc) and qualitatively (e.g. formally collecting user feedback on analytic fidelity and cybersickness).

An initial step towards developing a reproducible evaluation of the performance of a remote rendering implementation will be integrating profiling tools into the software. The current software provides instantaneous timing metrics for several of the aforementioned performance characterizations, but further work can be done to enable the collection of that data over time along with other related factors such as amount of user movement or measures of scene complexity. Additionally, a method to record timestamped user feedback on cybersickness would be valuable to cross-reference with the performance metrics.

A second step towards reproducible evaluation will be determining the variation caused by different scenes. A diverse set of scenes should be selected containing different situations that may adversely affect the success of remote rendering. Examples may include: Scenes with many large overlapping objects, scenes with a large field of tiny glyphs, scenes surrounding the viewer, monochromatic scenes, scenes with objects that traverse from the foreground to the background. As different types of scenes are found that provide varying results, they should be curated for iterative testing.

By performing these two steps, new techniques can be refined using these tools as a form of regression testing, as well as a means of uncovering unexpected opportunities for improvement. And by correlating qualitative user feedback on cybersickness with metrics such as missing pixels or content framerate, a better understanding of remote rendering's impact on cybersickness may arise.

As new methods are refined, it will be important to turn the focus towards user evaluations. Ultimately remote rendering serves as a tool for users in the absence of opportunities for locally rendered content. To make a case for the success of a remote rendering technique, it must be shown how similar a user's remote rendering experience is to a locally rendered experience, both in terms of cybersickness as well as utility for performing tasks. We can use methods for determining degrees of cybersickness, such as administering questionnaires, tracking postural sway, and evaluating physiological state [Rebenitsch and Owen 2016]. We can also measure impacts on spatial perception with approaches such as the experiments run by Aygar, Ware, and Rogers to assess the impact of stereoscopic imagery on a user's perception of the 3D structure of spatial datasets [2018].

7.6.2 Further Development

We expect that through testing and evaluation, many new refinements and approaches will emerge, but just from our initial research, we have several recommendations for next steps in software development.

Facade Generation Improvements

There are a number of areas where the DIBR technique described in this chapter can be optimized for boosts to the performance of this remote rendering implementation.

- **Perform some facade generation on the server:** We believe there may be benefit to moving much of the facade generation logic to the server, and packaging information about the facade mesh with the RGBD data. By not requiring the mesh deformation to occur on the client device, several of the following improvements may become more viable.

- **Floating-Point Depth:** Currently, the depth component of the RGBD image is 8-bits. In other words, the depth being used on the client side is split up into 255 evenly spaced bins between the eye and the farplane. By either transmitting higher-resolution depth information to the client or computing the mesh displacement on the server, the facade can be made smoother and have fewer artifacts such as the "stepping" seen in Figure 7.17 (right).

- **Object Edge Detection:** With more computational resources, objects that fall between the facade mesh vertices could be detected, ensuring that new quads can be more reliably injected for objects that appear at different depths.

- **Dynamic Mesh Resolution:** While detecting objects, the system could determine areas where a higher number of polygons are necessary, and determine other areas as candidates for mesh decimation. For example, regions with high-frequency depth details could be served by higher numbers of small quads, where large flat surfaces may need only a few quads.

- **Normal-Based Quad Alignment:** In Figure 7.12, note that the quads along object edges are view-aligned, whereas internal quads are more accurately aligned with the surface of the represented object due to all for corners being “pinned” by depth values. In the current implementation, injected quads along depth discontinuities don’t have depth samples for each corner, so some corners are just guessed from the average of the known corner depths. This results in sometimes noticeable “tag” shapes along the edge of objects. These artifacts could be alleviated by calculating a plane from the average normal of the object shown on the quad, and projecting the guessed corners onto that plane.

Compression Techniques

Currently, the only compression we’ve used for reducing the RGBD data is C#’s gzip library. A much more fitting approach would likely be to integrate techniques used for video streaming. We’ve already begun a collaboration with a researcher responsible for some cutting-edge work in 360 video streaming for VR [Qian et al. 2018]. We are interested in seeing how video streaming techniques can be integrated into Depth-Image Based Rendering.

One area to be considered is the necessity of lossless depth data. If the depth information is used to build a facade on the client and the depth or color information has been compressed in a lossy manner, it may result in the edges of objects bleeding into one another, and compression artifacts may manifest themselves spatially, perhaps impacting user comfort.

It may be found that video compression techniques will need to be modified to prioritize accuracy around regions of depth discontinuities.

Interaction Methods

Our implementation allows dynamic viewer-side reorientation of the remotely rendered content. This works because new content frames are requested relative to the Remote Rendering Client Prefab, to which the currently visible content frame facades are parented. So even though new content frames arrive with some amount of delay, they integrate smoothly into a dynamically moving view.

However, we do not presently support any user-driven changes to the actual server-side content. We do see benefit for many such interactions, such as moving cutting planes, adjusting streamline seed positions, or insertion of annotations. If we simply transmit user input events to the server, such actions will only appear after a delay at least equal to the Content Frame Fetch Delay. Such a delay may make it difficult to precisely interact with objects.

Thus, a system may be required to display a proxy of the changes introduced by the interaction while the viewer waits for the result of the interaction to be computed and remotely rendered. For example, the user may see a locally-rendered cutting plane indicator while adjusting its position, and the locally-rendered indicator would be hidden once the change is reflected in a new content frame.

A more general case to consider is the problem of showing remotely-rendered avatars of the user's hands or hand-held virtual tools. One solution that could be implemented with our existing tool would be to render the scene minus the hands (via layering) with one Remote Rendering Client Prefab, and then use another Remote Rendering Client Prefab parented to each hand, only requesting the corresponding hand imagery (again, via layering). On the Server side, A Remote Rendering Server Prefab could be similarly parented to the server-side hand objects that are being driven by hand-tracking events transmitted from the client. While visual changes to the hand avatars would be delayed, their movement would be locked to the user's tracked hand positions.

At this point it would seem that every type of interaction would need to be considered individually, but perhaps a generalized approach could be designed.

Collaboration Tools

As we consider a specific user's interactions, we also want to consider how interactions are presented to other collaborators who are simultaneously viewing the same scene.

A first step towards this would be incorporating and propagating user avatars to co-viewers. For example, user viewpoints can be shared with other viewers by showing a representation of their VR headset relative to the remotely rendered content. (It may then be worth considering limiting the degrees of freedom a user has for changing their viewpoint to avoid confusion!)

A next step would be integrating spot-lighting techniques, like laser-pointers or per-viewer regions of interest. These could either be applied directly to the remotely-rendered content (e.g. a colored flashlight used for projecting color onto the visualization on the server, to be captured by the remote rendering process), or independently rendered overlays (e.g. a 3D line showing the path of another user's pointer).

The design of good collaboration tools may be as complex a problem as actually developing their functionality. Many approaches have been demonstrated for different types of remote 2D collaboration indicators, such as Zoom whiteboards and Google Docs. For VR design collaboration, Unity has provided some examples of shared interaction techniques [Thomson 2019], but more research into viable techniques is recommended.

7.7 Conclusion

Just during the time it has taken to complete the work in this chapter, a number of industry leaders have made announcents regarding a number of remote-rendering related products. Amazon has announced Wavelength [Amazon 2020], a service that will use Amazon Web Services to offer remotely-rendered services to 5G devices, including VR/AR wearables. Nvidia has begun accepting application for accessing their new CloudXR SDK for creating AR and VR remote-rendering applications for 5G and WiFi streaming [Nvidia 2020]. And most recently, Qualcomm has announced its partnership with numerous other companies to build new 5G Edge-rendered AR and VR devices with its new processor over the next 5 years [Qualcomm 2020]. Clearly, remote rendering is just taking off, and the opportunities for remote-rendering research will only increase.

In light of all the work in the Remote Rendering space unfolding, we did not attempt to single-handedly solve the general problem. We instead focused on the specific problem of tackling cybersickness and analytic fidelity in the context of visualizing 3D scientific datasets. Moving forward, the techniques described in this chapter may mesh well with other upcoming advancements to build hybrid workflows that make data visualization a more collaborative and accessible resource to a wider audience.

Chapter 8

Conclusion

In this chapter, we present a final discussion of the dissertation, including a review of directions for ongoing and future research as well as a review of our key contributions and conclusions.

8.1 Discussion and Ongoing Work

The two works presented in chapters 5 - Chapter 7, Artifact-Based rendering and Remote Rendering, have come a long ways and are continuing to provide great utility in our ongoing research. In addition to enabling new types of research, we also see much potential in continued research and development upon these techniques themselves. In this section we will review and expand on the future work discussed in these latter chapters.

8.1.1 Artifact Based Rendering

As we completed our initial development for the Artifact-Based Rendering engine, we began welcoming more researchers into the team. Many of these ”Future Works” are already under development and show great promise as upcoming publications.

Advanced Artifact-Based Techniques and Style Inference

In our related work in Chapter 5, we referenced a number of artistic and non-photorealistic techniques that served as motivation for exploring artist-driven data visualizations. A natural next step would be to draw these tehcniques into the fold of what ABR has to offer.

As an additional example, Gorla, Interrante, and Sapiro show some inspiring work in synthesizing textures across arbitrary 3D surfaces, and even blend naturally between textures based on any function across the surface [2003]. Implementing such algorithms would provide an excellent foundation for data driven artifact-based rendering of isosurfaces.

As techniques like these further bolster ABR's flexibility, an exciting next step would be to refine a process of Style Inference. Similar to the 2D approach of "Image Analogies" [Hertzmann et al. 2001] where Hertzmann et al. can take the artistic style from one image and apply it to a totally different image, imagine a process where an artist could carefully create an Artifact-Based visualization of one dataset, and then that style could be lifted and applied as a starting point on a new dataset.

2D Design Interface

When we initially published Artifact-Based Rendering in 2019 [Johnson et al. 2019b], visualizations were designed through the Unity editor. This was never seen as ideal, as it required visualization designers to be familiar with Unity's workflow and opened up high potential for application-breaking user error. In 2020, our team began work on developing a browser-based design interface modeled specifically with artists in mind. The interface can even be accessed by remote participants, which pairs extremely well with Remote Rendering. Vis Assets can be drag-and-dropped from the online library, and connected to data variables and structural "key data" objects via interactive layer widgets. And states and screenshots can be saved and shared with other users. Even in its early iterations, it has provided a remarkable improvement in artist experience, and is currently undergoing initial tests with new artists.

For the immediate future, continued refinement will improve the browser-based design interface, but discussion is already happening for how best to integrate the design workflow into an immersive design experience either in VR or in an augmented-reality work space.

Immersive 3D Exploration Interface

Artifact-Based visualizations are designed to be experienced immersively. The shapes, textures, and colors drawn from the natural, physical world best facilitate human connection when a participant shares their space. Thus, an important set of questions revolve around what tasks that space should support and how the participant might interact with the space.

Some spatial interactions we are discussing are primarily focused on the composition and presentation of visualizations. We are experimenting with virtual lighting manipulation, and setting virtual viewpoints for capturing images and videos for publication. In the same way that we want to leverage artists' skills of painting, sculpting, and coloring through physical means, we also want to lower the barriers for visual designers to use their understanding of composition and lighting in as fluid and natural a manner as possible. Thus, we see VR as a prime platforms for making these sorts of design decisions.

VR also provides an excellent context for exploring data. We have begun work on incorporating a number of successful data interaction techniques. We've made initial strides towards implementing a virtual-reality version of Drawing with the Flow [Schroeder et al. 2010] in 3D, where a user can explore volumetric vector fields by sketching example illustrative streamlines, and refining them to accurately portray the underlying data. Methods like Interactive Slice World-In-Miniature [Coffey et al. 2012b] will be adapted to gain new perspectives on Artifact-Based visualizations. And ensemble visualization techniques like Bento Box [Johnson et al. 2019a] or Worlds-in-Wedges [Nam et al. 2019] are a natural framework into which ABR visualizations can be integrated.

Auto-Generated Legends

A critical feature for any truly useful scientific visualization is a meaningful legend. Systems like Paraview automatically display colormap legends showing how the color maps to the associated data variable (example in Figure 6.5). However, legends have been an ongoing challenge for ABR.

The challenge comes from the numerous possible encodings that can all be co-located on the same glyph, surface, or ribbon. Perhaps the most complex encoding in this paper are the brain glyphs seen in Figure 6.3. Note that in addition to being oriented based on brain fiber direction, the red-orange glyphs have data-driven aspect-ratio, color, and texture coming from three different variables. And in contrast, the blue glyphs have a double-encoding of color and aspect ratio for the same variable. A legend such as that in Figure 6.3(b) is useful, but also ultimately problematic in that any example glyphs used for one variable encoding, such as color, will inevitably also display possible representations for other encodings such as aspect ratio or texture. One could imagine a 3D grid of glyph examples, each of which demonstrating some combination of the three selected variables and their encodings. Perhaps in virtual reality such a multi-dimensional legend would be possible, but great care would need to be taken to ensure legibility.

In the meantime, a current investigation is being made into building automated 2D legends from any ABR visualization state. Claire Weissman, a fellow researcher in the Sculpting-Visualizations team, has recently prototyped what an automatically-generated legend might look like (Figure 8.1). She approaches the problem of these multi-variable encodings by using one encoding such as color to give a high-level visual description of the visualization, and supports drilling-down into the glyphs or textures used for each of these features.

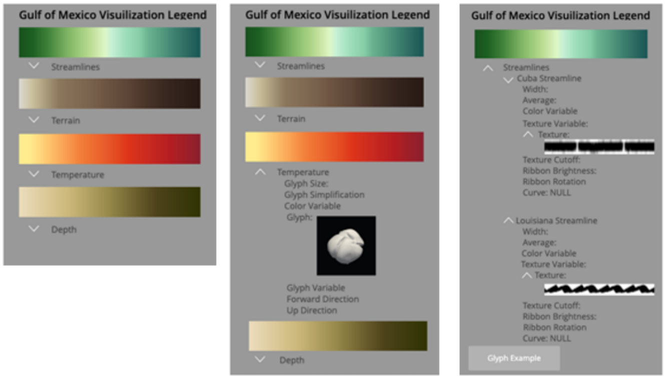


Figure 8.1: A prototype for what an automatically-generated legend could look like, with hierarchical drill-downs for seeing more detailed multi-variable encodings.

Measuring Affect

Another ongoing area of research for our team is quantifying the impact of the aesthetic of visualizations, and in particular how ABR affects the viewer, whether they are scientists or the public at large. Annie Bares, another researcher on our Sculpting Visualizations team, has recently published one work in how the approach of Close Reading can be used to understand how laypeople respond to visualization designs [Bares et al. 2020]. Work is currently being done in applying a similar approach for understanding affect for scientists.

8.1.2 Data Streaming & Remote Rendering

In Chapter 7, we provided detail on a number of next-steps for the Remote Rendering work. While our current implementation provides utility for accessing ABR visualizations on the Quest or Magic Leap and sharing visualizations between collaborators, Section 7.6.2 provides a road-map for several next-steps in improving the fidelity and reliability of the Remote Rendering experience.

Another related area our team has been moving towards is advancing our research into the Data Streaming process. Currently, our implementation pushes Paraview changes to an ABR application, but we currently don't provide and on-demand access to data. We plan to design a data server infrastructure that can host a number of different datasets, and from which specific variables and data objects can be requested based on needs of the visualization designer. A system like this will both equip artists to more fluidly apply their designs to different datasets, and enable more context-specific selection of data subsets to be transmitted, reducing the delay required to update a visualization. Such a server could even automatically apply sampling algorithms such as Metropolis-Hastings or similar density-based filters.

8.2 Summary and Review of Primary Contributions

In the introduction of this dissertation, we listed 4 overarching contributions. We review them here.

Several example works demonstrating the pillars of Palpable Visualizations.

With Bento Box in Chapter 3, we saw a number of ways in which accessibility and discernibility – our pillars of palpability – were showcased, as well as several areas where palpability could be improved. Motivated by our interactions with an artist during Bento Box, we stepped back and considered two art-based projects in Chapter 4, Weather Report and Lift-Off. We saw a number of new ways palpability was demonstrated through these creative processes, and we used the results to inform our theory of Artifact-Based Rendering in Chapter 5.

A theory and implementation of Artifact-Based Rendering, a framework of techniques for enabling artists to create Palpable Visualizations. In Chapter 5, we described the Artifact-Based rendering as a new means for creating visualizations with greater discernibility through artist-curated variety, providing those artists with a more accessible workflow for producing visualizations, and establishing a greater human connection between the data and a participant.

Early results of applying Artifact-Based Rendering to several diverse 3D datasets. In Chapter 6, we applied Artifact-Based Rendering to several datasets. With the bio-geochemistry in the Gulf of Mexico in particular, we performed an internal study in which we had an artist spend a day with a tool like Paraview, and compared the results with a day spent using ABR. Afterwards, the scientist responsible for the data provided feedback, remarking on the more engaging and detailed imagery he was excited to discover through the ABR approach.

An implementation and characterization of remote rendering from a high-performance rendering computer to a consumer untethered VR display with initial parallax correction techniques. Finally, in Chapter 7 we presented motivation and implementation details of our solution to the problem of remote collaboration via affordable, untethered VR displays, and wireless augmented reality devices. We provided an overview of the metrics that can be used to characterize the performance of remote rendering, and described our own internal demonstration of remote rendering as a platform for a collaborative remote design discussion.

8.3 Vision of the Future

The work in this dissertation focuses closely on technical hurdles necessary to enable the accessible and discernible experiences of palpable visualizations. In particular, we saw how advanced rendering and computational techniques can be combined and extended to produce visually rich and engaging data-driven imagery for a wide range of immersive headsets.

At this juncture, it would be appropriate to look ahead at what might compose the next dissertation-level research in Palpable Visualization. Now that the groundwork has been laid for how ABR can produce accessible artist-designed data visualizations, we need to consider both how to disseminate both these workflows as well as the resulting experiences. This could involve designing physical visualization-design benches that tightly link the scanning technology with AR displays of visualizations as they provide immediate feedback for the designer while they iteratively refine their artifacts by hand. For wide audiences, approaches to sharing these artifact-based results could be multi-faceted museum experiences involving hybrids between 3D printed physicalizations, augmented reality and virtual reality stations, and even planetarium experiences.

By focusing on the accessibility of wide-scale dissemination for both visualization creators and consumers, the next phase of Palpable Visualization could play a part in a new renaissance of engagement with art and science.

8.4 General Conclusions

As we conclude, let us re-visit our thesis statement:

Interactive computational workflows that equip artists to fluidly design scientific visualizations and enable any participant to immersively experience and relate with the results can lead to visualizations that are simultaneously more discernible and more accessible.

This research explores how the human connection to science can be enhanced through leveraging the familiarity and variation of the physical world. By inviting artists into our workflows, our data, and our software through *fluid design* interfaces, we begin to tap into not just a new aesthetic for our visualizations, but an entirely new paradigm of visual communication with an emphasis on *relating* physically with science.

Bento Box showed us that reaching out with our hands provides a natural and fluid method of interacting with and understanding an actively studied ensemble of ten engineering simulations, a massive challenge for rendering algorithms. But it also revealed that our software design precluded the involvement of artists in our visualization design process. This provides evidence that involving artists in the design of high-end scientific visualization is useful; however, it also demonstrates that current visualization practices limit their involvement.

In contrast, Weather Report turned to professional artists and architects to design and construct a physical, touchable representation of weather data and the result was far more approachable, reaching hundreds of participants from the public. This provides evidence that artists can help us engage the public with science; however, the presentation prioritized accessibility over discernibility. This type of public artwork does not fully solve the problem of making high-end scientific visualization both more discernible and accessible. As a next step in bringing artistic and creative tools into the scientific domain, Lift-Off showed how we can begin to model our scientific procedures on the physical creative processes that we all have learned, like napkin sketching and gestures in the air. This provides evidence that traditional design methods like sketching have a role to play in science; however, the applications here did not focus on multi-variable visualization problems.

Bringing together these lessons in palpable discernment and accessibility, Artifact-Based Rendering provides a *fluid visualization design process* and the display of data through physical artifacts. And we see success as we recall the feedback from our artists and scientists. Our artists praised the ABR design workflow as giving her *access* to as wide an array of design possibilities as her physical art studio, and one of our scientists was nearly taken aback at how approachable and engaging the aesthetic was. And most importantly, we've balanced this accessibility with discernibility. Thanks to the extra channels of information we can encode with shape and texture in our ABR VisLayers, that scientist found these visualizations to be transformational, letting him *discern* combinations of variables he couldn't otherwise.

And with our Remote Rendering techniques, we can share ABR visualizations driven by huge super-computed datasets with remote audiences using a wide range of immersive devices to allow scientists and the public to *immersively experience* data.

As we look to the future, we see an explosion of rich, interdisciplinary collaborations, breaking down the technical and psychological divides between art and science. We see Artifact-Based Rendering as one large step towards bringing scientific data into the artist's palette, perceptually accurate visual design philosophies into the scientist's tool belt, and a deeper understanding of complex scientific research to the fingertips of everyone.

References

Maneesh Agrawala and Chris Stolte. 2001. Rendering Effective Route Maps: Improving Usability Through Generalization. In *Proceedings of the 28th annual conference on Computer Graphics and Interactive Techniques*. ACM, 241–249.

James Ahrens, Berk Geveci, and Charles Law. 2005. Paraview: An end-user tool for large data visualization. *The visualization handbook* 717 (2005).

Oluwafemi S Alabi, Xunlei Wu, Jonathan M Harter, Madhura Phadke, Lifford Pinto, Hannah Petersen, Steffen Bass, Michael Keifer, Sharon Zhong, Chris Healey, et al. 2012. Comparative visualization of ensembles using ensemble surface slicing. In *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 82940U–82940U.

J. Albers. 2009. *The Interaction of Color.* Yale University Press, New Haven, CT.

Jason Alexander, Yvonne Jansen, Kasper Hornbæk, Johan Kildal, and Abhijit Karnik. 2015. Exploring the challenges of making data physical. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 2417–2420.

Amazon. 2020. *Amazon Wavelength.* `https://aws.amazon.com/wavelength/`

Erol Aygar, Colin Ware, and David Rogers. 2018. The contribution of stereoscopic and motion depth cues to the perception of structures in 3D point clouds. *ACM Transactions on Applied Perception (TAP)* 15, 2 (2018), 1–13.

Annie Bares, Daniel F. Keefe, and Francesca Samsel. 2020. Close Reading for Visualization Evaluation. 40, 04 (July 2020), 84–95.

Štěpán Beneš and Jaroslav Kruis. 2015. Efficient methods to visualize finite element meshes. *Advances in Engineering Software* 79 (2015), 81–90.

Dan S. Bloomberg. 2008. Color quantization using octrees.

Ralf P Botchen, Daniel Weiskopf, and Thomas Ertl. 2005. Texture-based visualization of uncertainty in flow fields. In *VIS 05. IEEE Visualization, 2005.* IEEE, 647–654.

Doug Bowman, Ernst Kruijff, Joseph J LaViola Jr, and Ivan P Poupyrev. 2004. *3D User interfaces: theory and practice, CourseSmart eTextbook.* Addison-Wesley.

Doug A Bowman and Larry F Hodges. 1997. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics.* ACM, 35–ff.

Stefan Bruckner and Eduard Gröller. 2007. Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1344–1351.

Stefan Bruckner and M Eduard Groller. 2005. *Volumeshop: An interactive system for direct volume illustration.* IEEE.

Stefan Bruckner and Torsten Moller. 2010. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (2010), 1468–1476.

Steve Bryson. 1996. Virtual reality in scientific visualization. *Commun. ACM* 39, 5 (1996), 62–71.

R. Bujack, T. L. Turton, F. Samsel, C. Ware, D. H. Rogers, and J. Ahrens. 2018. The Good, the Bad, and the Ugly: A Theoretical Framework for the Assessment of Continuous Colormaps. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan. 2018), 923–933. `https://doi.org/10.1109/TVCG.2017.2743978`

Jeff Butterworth, Andrew Davidson, Stephen Hench, and Marc. T. Olano. 1992. 3DM: A Three Dimensional Modeler Using a Head-mounted Display. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics.* ACM, New York, NY, USA, 135–138.

Bill Buxton. 2010. *Sketching user experiences: getting the design right and the right design.* Morgan kaufmann.

John Carmack. 2013. Latency Mitigation Strategies. `https://web.archive.org/web/20140719085135/http://altdev.co/2013/02/22/latency-mitigation-strategies/`

John Carmack. 2019. Oculus Connect 6 - Day 2 Keynote. (2019). `https://youtu.be/PMIDaomx0GA` Oculus Connect 6.

E. H. . Chi, P. Barry, J. Riedl, and J. Konstan. 1997. A spreadsheet approach to information visualization. In *Proceedings of VIZ '97: Visualization Conference, Information Visualization Symposium and Parallel Rendering Symposium.* 17–24. `https://doi.org/10.1109/INFVIS.1997.636761`

Siddhartha Chib and Edward Greenberg. 1995. Understanding the metropolis-hastings algorithm. *The american statistician* 49, 4 (1995), 327–335.

Dane Coffey, Fedor Korsakov, Marcus Ewert, Haleh Hagh-Shenas, Lauren Thorson, A Ellingson, D Nuckley, and Daniel F Keefe. 2012a. Visualizing motion data in virtual reality: Understanding the roles of animation, interaction, and static presentation. 31, 3pt3 (2012), 1215–1224.

Dane Coffey, Chi-Lun Lin, Arthur G Erdman, and Daniel F Keefe. 2013. Design by dragging: An interface for creative forward and inverse design with simulation ensembles. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2783–2791.

Dane Coffey, Nicholas Malbraaten, Trung Le, Iman Borazjani, Fotis Sotiropoulos, Arthur G. Erdman, and Daniel F. Keefe. 2012b. Interactive Slice WIM: Navigating and Interrogating Volume Datasets Using a Multi-Surface, Multi-Touch VR Interface. *IEEE Transactions on Visualization and Computer Graphics* 18, 10 (2012), 1614–1626. `https://doi.org/10.1109/TVCG.2011.283`

Dane Coffey, Nicholas Malbraaten, Trung Bao Le, Iman Borazjani, Fotis Sotiropoulos, Arthur G Erdman, and Daniel F Keefe. 2012c. Interactive slice WIM: Navigating and interrogating volume data sets using a multisurface, multitouch VR interface. *IEEE Transactions on Visualization and Computer Graphics* 18, 10 (2012), 1614–1626.

Donna J Cox. 1988. Using the Supercomputer to Visualize Higher Dimensions: An Artist's Contribution to Scientific Visualization. *Leonardo* 21, 3 (1988), 233–242.

Donna J Cox. 2006. Metaphoric mappings: The art of visualization. *Aesthetic computing* (2006), 89–114.

Donna J Cox. 2008. Using the supercomputer to visualize higher dimensions: An artist's contribution to scientific visualization. *Leonardo* 41, 4 (2008), 391–400.

Lawrence D Cutler, Bernd Fröhlich, and Pat Hanrahan. 1997. Two-handed direct manipulation on the responsive workbench. In *Proceedings of the 1997 symposium on Interactive 3D graphics*. ACM, 107–114.

Michael F. Deering. 1995. HoloSketch: A Virtual Reality Sketching/Animation Tool. *ACM Trans. on Computer-Human Interaction* 2, 3 (1995), 220–238.

Hessam Djavaherpour, Ali Mahdavi-Amiri, and Faramarz F Samavati. 2017. Physical visualization of geospatial datasets. *IEEE computer graphics and applications* 37, 3 (2017), 61–69.

Klaus Engel and Thomas Ertl. 1999. Texture-based volume visualization for multiple users on the world wide web. In *Virtual Environments' 99*. Springer, 115–124.

Klaus Engel, Roberto Grosso, and Thomas Ertl. 1998. Progressive isosurfaces on the web. *Late Breaking Hot Topics Proc. Visualization* 98 (1998).

Klaus Engel, Ove Sommer, Christian Ernst, and Thomas Ertl. 1999. Remote 3d visualization using image-streaming techniques. In *ISIMADE-11 TH Internationl Conference on Systems Research, Informatics and Cybernetics.*

P. Evans and N. Thomas. 2008. *The Elements of Design.* Cengage Learning, Clifton Park, NY.

Martin Falk, Sebastian Grottel, Michael Krone, and Guido Reina. 2016. Interactive GPU-based visualization of large dynamic particle data. *Synthesis Lectures on Visualization* 4, 3 (2016), 1–121.

Hamza Farooq, Junqian Xu, Jung Who Nam, Daniel F Keefe, Essa Yacoub, Tryphon Georgiou, and Christophe Lenglet. 2016a. Microstructure imaging of crossing (MIX) white matter fibers from diffusion MRI. *Scientific reports* 6 (2016), 38927.

Hamza Farooq, Junqian Xu, Essa Yacoub, Tryphon Georgiou, and Christophe Lenglet. 2016b. Brain Tissue Micro-Structure Imaging from Diffusion MRI Using Least Squares Variable Separation. In *Computational Diffusion MRI*. Springer, 55–64.

David Feng, Yueh Lee, Lester Kwock, and Russell M Taylor II. 2009. Evaluation of Glyph-based Multivariate Scalar Volume Visualization Techniques. In *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization*. ACM, 61–68.

Danyel Fisher, Igor Popov, Steven Drucker, et al. 2012. Trust me, I'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1673–1682.

Jerrold A Friesen and Thomas D Tarman. 2000. Remote high-performance visualization and collaboration. *IEEE Computer Graphics and Applications* 20, 4 (2000), 45–49.

Johannes Fuchs, Petra Isenberg, Anastasia Bezerianos, and Daniel Keim. 2017. A Systematic Review of Experimental Studies on Data Glyphs. *IEEE Transactions on Visualization and Computer Graphics* 23, 7 (2017), 1863–1879.

Andrew Gardner, Chris Tchou, Tim Hawkins, and Paul Debevec. 2003. Linear light source reflectometry. *ACM Transactions on Graphics* 22, 3 (2003), 749–758.

Christoph Garth and Kenneth I Joy. 2010. Fast, memory-efficient cell location in unstructured grids for visualization. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (2010), 1541–1550.

Michael Gleicher. 2017. Considerations for Visualizing Comparison. *IEEE transactions on visualization and computer graphics* (Aug. 2017).

Michael Gleicher, Danielle Albers, Rick Walker, Ilir Jusufi, Charles D Hansen, and Jonathan C Roberts. 2011. Visual comparison for information visualization. *Information Visualization* 10, 4 (2011), 289–309.

Michael Glueck, Azam Khan, and Daniel J Wigdor. 2014. Dive in!: Enabling progressive loading for real-time navigation of data visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 561–570.

Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. 1998. A Non-photorealistic Lighting Model for Automatic Technical Illustration. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 447–452.

Gabriele Gorla, Victoria Interrante, and Guillermo Sapiro. 2003. Texture synthesis for 3D shape representation. *IEEE Transactions on Visualization and Computer Graphics* 9, 4 (2003), 512–524.

Hanqi Guo, Ningyu Mao, and Xiaoru Yuan. 2011. Wysiwyg (what you see is what you get) volume visualization. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2106–2114.

Christopher G Healey. 2001. Formalizing artistic techniques and scientific visualization for painted renditions of complex information spaces. In *IJCAI*. 371–376.

Christopher G Healey and James T Enns. 1999. Large datasets at a glance: Combining textures and colors in scientific visualization. *IEEE transactions on visualization and computer graphics* 5, 2 (1999), 145–167.

Christopher G Healey and James T Enns. 2002. Perception and Painting: A Search for Effective, Engaging Visualizations. *IEEE Computer Graphics and Applications* 22, 2 (2002), 10–15.

Paul Heckbert. 1982. Color image quantization for frame buffer display. *ACM SIGGRAPH* 16, 3 (1982), 297–307.

Paul S Heckbert. 1986. Survey of texture mapping. *IEEE computer graphics and applications* 6, 11 (1986), 56–67.

Bernd Hentschel, Irene Tedjo, Markus Probst, Marc Wolter, Marek Behr, Christian Bischof, and Torsten Kuhlen. 2008. Interactive blood damage analysis for ventricular assist devices. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008), 1515–1522.

Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. 2001. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 327–340.

Ken Hinckley, Randy Pausch, Dennis Proffitt, James Patten, and Neal Kassell. 1997. Cooperative bimanual action. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*. ACM, 27–34.

Trevor Hogan and Eva Hornecker. 2016. Towards a design space for multisensory data representation. *Interacting with Computers* 29, 2 (2016), 147–167.

V. Interrante. 2000. Harnessing natural textures for multivariate visualization. *IEEE Computer Graphics and Applications* 20, 6 (Nov. 2000), 6–11. `https://doi.org/10.1109/MCG.2000.888001`

V. Interrante and C. Grosch. 1997. Strategies for effectively visualizing 3D flow with volume LIC. In *Proceedings of the 8th conference on Visualization*. IEEE Computer Society Press, 421–424. `https://doi.org/10.1109/VISUAL.1997.663912`

J. Itten and F. Birren. 1970. *The Elements of Color: A Treatise on the Color System of Johannes Itten Based on His Book the Art of Color* (1st ed.). Van Nostrand Reinhold Company, New York, NY.

Harry W. Bullen IV, Jessica S. Chang, Alexander V. Harn, Sean P. Kelly, Steven G. Satterfield, Peter M. Ketcham, and Judith E. Devaney. 2002. A Glyph Toolbox for Immersive Scientific Visualization.

Bret Jackson. 2017. *MinVR*. `https://github.com/MinVR/MinVR`

Bret Jackson and Daniel F. Keefe. 2016. Lift-Off: Using Reference Imagery and Freehand Sketching to Create 3D Models in VR. *IEEE Transactions on Visualization and Computer Graphics* 22, 4 (April 2016), 1442–1451.

TJ Jankun-Kelly and Kwan-Liu Ma. 2001. Visualization exploration and encapsulation via a spreadsheet-like interface. *IEEE Transactions on Visualization and Computer Graphics* 7, 3 (2001), 275–287.

Yvonne Jansen, Pierre Dragicevic, Petra Isenberg, Jason Alexander, Abhijit Karnik, Johan Kildal, Sriram Subramanian, and Kasper Hornbæk. 2015. Opportunities and challenges for data physicalization. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 3227–3236.

Waqas Javed and Niklas Elmqvist. 2010. Stack zooming for multi-focus interaction in time-series data visualization. In *Visualization Symposium (PacificVis), 2010 IEEE Pacific*. IEEE, 33–40.

Waqas Javed, Sohaib Ghani, and Niklas Elmqvist. 2012. Polyzoom: multiscale and multifocus exploration in 2d visual spaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 287–296.

C. Johnson. 2004. Top scientific visualization research problems. *IEEE Computer Graphics and Applications* 24, 4 (July 2004), 13–17. `https://doi.org/10.1109/MCG.2004.20`

Seth Johnson, Bret Jackson, Bethany Tourek, Marcos Molina, Arthur G Erdman, and Daniel F Keefe. 2016. Immersive analytics for medicine: hybrid 2D/3D sketch-based interfaces for annotating medical data and designing medical devices. In *Proceedings of the 2016 ACM Companion on Interactive Surfaces and Spaces*. 107–113.

Seth Johnson, Daniel Orban, Hakizumwami Birali Runesha, Lingyu Meng, Bethany Juhnke, Arthur Erdman, Francesca Samsel, and Daniel F Keefe. 2019a. Bento Box: An Interactive and Zoomable Small Multiples Technique for Visualizing 4D Simulation Ensembles in Virtual Reality. *Front. Robot. AI 6: 61. doi: 10.3389/frobt* (2019).

Seth Johnson, Francesca Samsel, Gregory Abram, Daniel Olson, Andrew J Solis, Bridger Herman, Phillip J Wolfram, Christophe Lenglet, and Daniel F Keefe. 2019b. Artifact-Based Rendering: Harnessing Natural and Traditional Visual Media for More Expressive and Engaging 3D Visualizations. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 492–502.

Alark Joshi and Penny Rheingans. 2005. Illustration-inspired techniques for visualizing time-varying data. In *Visualization, 2005. VIS 05. IEEE*. IEEE, 679–686.

Ralf Kähler, Donna Cox, Robert Patterson, Stuart Levy, Hans Christian Hege, and Tom Abel. 2002. Rendering the First Star in the Universe: A Case Study. In *Proceedings of IEEE Visualization*. IEEE, 537–540.

Robert D Kalnins, Lee Markosian, Barbara J Meier, Michael A Kowalski, Joseph C Lee, Philip L Davidson, Matthew Webb, John F Hughes, and Adam Finkelstein. 2002. WYSIWYG NPR: Drawing strokes directly on 3D models. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 755–762.

Daniel F Keefe, Daniel Acevedo, Jadrian Miles, Fritz Drury, Sharon M Swartz, and David H Laidlaw. 2008. Scientific Sketching for Collaborative VR Visualization Design. *IEEE Transactions on Visualization and Computer Graphics* 14, 4 (2008), 835–847.

Daniel F Keefe, Daniel Acevedo Feliz, Tomer Moscovich, David H Laidlaw, and Joseph J LaViola Jr. 2001. CavePainting: a fully immersive 3D artistic medium and interactive experience. In *Proceedings of the 2001 symposium on Interactive 3D graphics*. ACM, 85–93.

Daniel F Keefe, Seth Johnson, Ross Altheimer, Deuk-Geun Hong, Robert Hunter, Andrea J Johnson, Maura Rockcastle, Mark Swackhamer, and Aaron Wittkamper. 2018. Weather Report: A Site-Specific Artwork Interweaving Human Experiences and Scientific Data Physicalization. *IEEE Computer Graphics and Applications* 38, 4 (2018), 10–16.

Daniel F Keefe, David B Karelitz, Eileen L Vote, and David H Laidlaw. 2005. Artistic collaboration in designing VR visualizations. *IEEE Computer Graphics and Applications* 25, 2 (2005), 18–23.

Daniel F Keefe, Robert C Zeleznik, and David H Laidlaw. 2007. Drawing on air: Input techniques for controlled 3D line illustration. *IEEE Transactions on Visualization and Computer Graphics* 13, 5 (2007), 1067–1081.

Johannes Kehrer and Helwig Hauser. 2013. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE transactions on visualization and computer graphics* 19, 3 (2013), 495–513.

Rohit Ashok Khot, Larissa Hjorth, and Florian ‘Floyd’ Mueller. 2014. Understanding physical activity through 3D printed material artifacts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3835–3844.

Kyungyoon Kim, John V. Carlis, and Daniel F. Keefe. 2017. Comparison techniques utilized in spatial 3D and 4D data visualizations: A survey and future directions. 67, Supplement C (2017), 138–147.

Gordon Kindlmann. 2004. Superquadric Tensor Glyphs. In *Proceedings of the Sixth Joint Eurographics-IEEE TCVG conference on Visualization*. Eurographics Association, 147–154.

Robert M Kirby, Daniel F Keefe, and David H Laidlaw. 2005. Painting and Visualization. *The Visualization Handbook* (2005), 873–891.

Robert M Kirby, Haralambos Marmanis, and David H Laidlaw. 1999. Visualizing multivalued data from 2D incompressible flows using concepts from painting. In *Proceedings of the conference on Visualization'99: celebrating ten years*. IEEE Computer Society Press, 333–340.

Roberta L Klatzky and Joann Peck. 2012. Please touch: Object properties that invite touch. *IEEE Transactions on Haptics* 5, 2 (2012), 139–147.

P. Kovesi. 2014. *Good Colour Maps: How to Design Them.* arXiv preprint arXiv:1509.03700. `http://peterkovesi.com/projects/colourmaps/ColourmapTheory/`

Falko Kuester, Ralph Bruckschen, Bernd Hamann, and Kenneth I. Joy. 2001. Visualization of particle traces in virtual environments. In *VRST*.

Bireswar Laha and Doug A Bowman. 2013. Volume cracker: a bimanual 3D interaction technique for analysis of raw volumetric data. In *Proceedings of the 1st symposium on Spatial user interaction*. ACM, 61–68.

Zeqi Lai, Y Charlie Hu, Yong Cui, Linhui Sun, Ningwei Dai, and Hung-Sheng Lee. 2019. Furion: Engineering high-quality immersive virtual reality on today's mobile devices. *IEEE Transactions on Mobile Computing* (2019).

David H Laidlaw, Eric T Ahrens, David Kremers, Matthew J Avalos, Russell E Jacobs, and Carol Readhead. 1998. Visualizing diffusion tensor images of the mouse spinal cord. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*. IEEE, 127–134.

D. Lauer and S. Pentar. 2012. *Design Basics*. Wadsworth, Boston, MA.

Joseph J LaViola Jr. 2000. A discussion of cybersickness in virtual environments. *ACM Sigchi Bulletin* 32, 1 (2000), 47–56.

Bongshin Lee, Rubaiat Habib Kazi, and Greg Smith. 2013. SketchStory: Telling more engaging stories with data through freeform sketching. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2416–2425.

Eun-Jin Lee and Sherif El-Tawil. 2008. FEMvrml: An interactive virtual environment for visualization of finite element simulation results. *Advances in Engineering Software* 39, 9 (2008), 737–742.

Andrea Leganchuk, Shumin Zhai, and William Buxton. 1998. Manual and cognitive benefits of two-handed input: an experimental study. *ACM Transactions on Computer-Human Interaction (TOCHI)* 5, 4 (1998), 326–359.

P. A. Legg, E. Maguire, S. Walton, and M. Chen. 2017. Glyph Visualization: A Fail-Safe Design Scheme Based on Quasi-Hamming Distances. *IEEE Transactions on Computer Graphics and Applications* 37, 2 (2017), 31–41.

Xu Liangyin, Li Yunpeng, Zhang Sheng, and Chen Biaosong. 2018. Efficient visualization strategies for large-scale finite element models. *Journal of Computing and Information Science in Engineering* 18, 1 (2018), 011007.

Santiago V. Lombeyda. 2016. Distinct 3D Glyphs with Data Layering for Highly Dense Multivariate Data Plots. *arXiv Preprint* (2016).

Xin Lu, Poonam Suryanarayan, Reginald B Adams Jr., Jia Li, Michelle G Newman, and James Z Wang. 2012. On shape and the computability of emotions. In *Proceedings of the 20th ACM international conference on Multimedia*. ACM, 229.

Jonas Lukasczyk, Eric Kinner, James Ahrens, Heike Leitte, and Christoph Garth. 2018. Voidga: A view-approximation oriented image database generation approach. In *2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)*. 12–22.

Eric Jason Luke and Charles D Hansen. 2002. *Semotus visum: a flexible remote visualization framework.* IEEE.

Giorgia Lupi. 2017. Data Humanism: The Revolutionary Future of Data Visualization. *Print Magazine* (Jan. 2017).

Giorgia Lupi, Stefanie Posavec, and Maria Popova. 2016. *Dear data.* Princeton Architectural Press.

Kwan-Liu Ma, Aaron Hertzmann, Victoria Interrante, and Eric B Lum. 2002. Recent advances in non-photorealistic rendering for art and visualization. *SIGGRAPH 2002 Course Notes. Course* 23 (2002).

Daniel P Mapes and J Michael Moshell. 1995. A two-handed interface for object manipulation in virtual environments. *Presence: Teleoperators & Virtual Environments* 4, 4 (1995), 403–416.

Peter Mitchell, Colin Ware, and John Kelley. 2009. Investigating flow visualizations using interactive design space hill climbing. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on.* IEEE, 355–361.

J Keith Moore, Scott C Doney, Joanie A Kleypas, David M Glover, and Inez Y Fung. 2001. An intermediate complexity marine ecosystem model for the global domain. *Deep Sea Research Part II: Topical Studies in Oceanography* 49, 1-3 (2001), 403–462.

J Keith Moore, Keith Lindsay, Scott C Doney, Matthew C Long, and Kazuhiro Misumi. 2013. Marine ecosystem dynamics and biogeochemical cycling in the Community Earth System Model [CESM1 (BGC)]: Comparison of the 1990s with the 2090s under the RCP4. 5 and RCP8. 5 scenarios. *Journal of Climate* 26, 23 (2013), 9291–9312.

K. Moreland. 2009. Diverging Color Maps for Scientific Visualization. In *Proceedings of the 5th International Symposium on Advances in Visual Computing, Part II (ISVC '09)*. 92–103.

Tamara Munzner. 2014. *Visualization analysis and design.* CRC Press.

Oleg Muratov, Yury Slynko, Vitaly Chernov, Maria Lyubimtseva, Artem Shamsuarov, and Victor Bucha. 2016. 3DCapture: 3D Reconstruction for a Smartphone. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.* 75–82.

Jung Who Nam, Krista McCullough, Joshua Tveite, Maria Molina Espinosa, Charles H Perry, Barry T Wilson, and Daniel F Keefe. 2019. Worlds-in-Wedges: Combining Worlds-in-Miniature and Portals to Support Comparative Immersive Visualization of Forestry Data. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR).* IEEE, 747–755.

Engineering National Academies of Sciences, David Skorton Medicine, and Ashley Bear editors. 2018. *The Integration of the Humanities and Arts with Sciences, Engineering, and Medicine in Higher Education: Branches from the Same Tree.* The National Academies Press, Washington, DC. `https://doi.org/10.17226/24988`

NorthernLights.mn. 2016. *Northern Spark Call for Proposals: Climate Chaos — Climate Rising.* `http://www.mnartists.org/content/northern-spark-call-proposals-climate-chaos-climate-rising`

Wieslaw L Nowinski, Anthony Fang, Bonnie T Nguyen, Jose K Raphel, Lakshmipathy Jagannathan, Raghu Raghavan, R Nick Bryan, and Gerald A Miller. 1997. Multiple brain atlas database and atlas-based neuroimaging system. *Computer Aided Surgery* 2, 1 (1997), 42–66.

Nvidia. 2020. *NVIDIA CloudXR™ SDK.* `https://developer.nvidia.com/nvidia-cloudxr-sdk`

Harald Obermaier and Kenneth I Joy. 2014. Future challenges for ensemble visualization. *IEEE Computer Graphics and Applications* 34, 3 (2014), 8–11.

Oculus. [n.d.]. *Testing and Performance Analysis.* `https://developer.oculus.com/documentation/unity/unity-perf`

Manuel M Oliveira, Gary Bishop, and David McAllister. 2000. Relief texture mapping. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 359–368.

J Logan Olson, David M Krum, Evan A Suma, and Mark Bolas. 2011. A design for a smartphone-based head mounted display. In *2011 IEEE Virtual Reality Conference*. IEEE, 233–234.

John M. Patchett, Francesca Samsel, Karen C. Tsai, Galen R. Gisler, David Rogers, Greg Abram, and Terece L. Turton. 2016. Visualization and Analysis of Threats from Asteroid Ocean Impacts. In *Proceedings of the 2016 ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis (SC)*. IEEE. Winner, Best Scientific Visualization & Data Analytics Showcase.

Mark R Petersen, Douglas W Jacobsen, Todd D Ringler, Matthew W Hecht, and Mathew E Maltrud. 2015. Evaluation of the arbitrary Lagrangian–Eulerian vertical coordinate method in the MPAS-Ocean model. *Ocean Modelling* 86 (2015), 93–113.

Polygraphene. 2019. ALVR - Air Light VR. `https://github.com/polygraphene/ALVR`.

Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. 2018. Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM, 99–114.

Qualcomm. 2020. *Qualcomm Collaborates with 15 Global Operators to Deliver XR Viewers*. `http://www.qualcomm.com/news/releases/2020/05/26/qualcomm-collaborates-15-global-operators-deliver-xr-viewers`

Mohammad Raji, Alok Hota, Tanner Hobson, and Jian Huang. 2018. Scientific Visualization as a Microservice. *IEEE transactions on visualization and computer graphics* (2018).

Mohammad Raji, Alok Hota, and Jian Huang. 2017. Scalable web-embedded volume rendering. In *2017 IEEE 7th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, 45–54.

Lisa Rebenitsch and Charles Owen. 2016. Review on cybersickness in applications and visual displays. *Virtual Reality* 20, 2 (2016), 101–125.

Penny L Rheingans. 2000. Task-based color scale design. In *Proceedings of the 28th AIPR Workshop: 3D Visualization for Data Exploration and Decision Making*, Vol. 3905. International Society for Optics and Photonics, 35–44.

Theresa Marie Rhyne. 2016. *Applying Color Theory to Digital Media and Visualization.* CRC Press.

Todd Ringler, Mark Petersen, Robert L Higdon, Doug Jacobsen, Philip W Jones, and Mathew Maltrud. 2013. A multi-resolution approach to global ocean modeling. *Ocean Modelling* 69 (2013), 211–232.

George Robertson, Roland Fernandez, Danyel Fisher, Bongshin Lee, and John Stasko. 2008. Effectiveness of animation in trend visualization. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008).

David H. Rogers, Francesca Samsel, Daniel F. Keefe, Benjamin Bach, and Lyn Bartram. 2017. Discovery Jam. In *IEEE VIS, Accepted Workshop.*

David H. Rogers, Francesca Samsel, Daniel F. Keefe, Mariah Meyer, Cecilia Aragon, Nina McCurdy, and Ethan Kerzner. 2016. Discovery Jam. In *IEEE VIS, Accepted Workshop.*

B. E. Rogowitz and A. D. Kalvin. 2001. The "Which Blair Project": A Quick Visual Method for Evaluating Perceptual Color Maps. In *Proceedings Visualization 2001 (VIS'01).* 183ff.

Timo Ropinski, Steffen Oeltze, and Bernhard Preim. 2011. Survey of glyph-based visualization techniques for spatial multivariate medical data. *Computers & Graphics* 35, 2 (2011), 392–401.

Hakizumwami Birali Runesha, Bogdan Florin Tanasoiu, Georgi Subashki, Arthur Erdman, and Daniel Keefe. 2016. Podium Presentation: Fluid-Structure Interaction Simulation of Cardiac Leads in the Heart: Developing a Computational Model for use in Medical Device Design. Design of Medical Devices Conference.

Francesca Samsel. 2013. Art-Science-Visualization Collaborations: Examining the Spectrum. In *Proceedings of the IEEE VIS Arts Program (VISAP)*. IEEE.

Francesca Samsel, Lyn Bartram, Annie, and Bares. 2018. Art, Affect and Color: Creating Engaging Expressive Scientific Visualization. In *Proceedings of IEEE Visualization.*

Francesca Samsel and Daniel F. Keefe. 2013. Meet the Scientists. In *College Art Association Annual Meeting.*

Francesca Samsel, Sebastian Klaassen, Mark Petersen, Terece L Turton, Gregory Abram, David H Rogers, and James Ahrens. 2016. Interactive Colormapping: Enabling Multiple Data Range and Detailed Views of Ocean Salinity. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 700–709.

F. Samsel, M. Petersen, T. Turton, G. Abram, J. Wendelberger, and J. Ahrens. 2015. Colormaps That Improve Perception of High-Resolution Ocean Data. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '15)*. Seoul, Republic of Korea, 703–710. `https://doi.org/10.1145/2702613.2702975`

Purvi Saraiya, Chris North, and Karen Duca. 2005. An insight-based methodology for evaluating bioinformatics visualizations. *IEEE transactions on visualization and computer graphics* 11, 4 (2005), 443–456.

Richard M Satava. 1993. Virtual reality surgical simulator. *Surgical endoscopy* 7, 3 (1993), 203–205.

Gernot Schaufler and Wolfgang Stürzlinger. 1996. A three dimensional image cache for virtual reality. In *Computer Graphics Forum*, Vol. 15. Wiley Online Library, 227–235.

Ben Schneiderman. 2007. Creativity support tools: Accelerating discovery and innovation. *Commun. ACM* 50 (2007), 20–32. Issue 12.

Arno Schödl, Richard Szeliski, David H Salesin, and Irfan Essa. 2000. Video textures. In *Proceedings of the 27th annual conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co., 489–498.

David Schroeder, Dane Coffey, and Daniel F Keefe. 2010. Drawing with the flow: A sketch-based interface for illustrative visualization of 2D vector fields. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*. Eurographics Association, 49–56.

D. Schroeder and D. F. Keefe. 2016. Visualization-by-Sketching: An Artist's Interface for Creating Multivariate Time-Varying Data Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (Jan. 2016), 877–885. `https://doi.org/10.1109/TVCG.2015.2467153`

Will J Schroeder, Bill Lorensen, and Ken Martin. 2004. *The visualization toolkit: an object-oriented approach to 3D graphics*. Kitware.

Michael Sedlmair, Christoph Heinzl, Stefan Bruckner, Harald Piringer, and Torsten Möller. 2014. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2161–2170.

Michael Sedlmair, Petra Isenberg, Miriah Meyer, and Tobias Isenberg. 2018. The 7th biennial BELIV workshop, BELIV 2018: evaluation and Beyond - methodoLogIcal approaches for Visualization. In *IEEE VIS, Accepted Workshop*.

Adrien Sengal. 2015. Grewingk Glacier. https://www.adriensegal.com/grewingk-glacier. Accessed March 2019.

François Sillion, George Drettakis, and Benoit Bodelet. 1997. Efficient impostor manipulation for real-time visualization of urban scenery. In *Computer Graphics Forum*, Vol. 16. Wiley Online Library, C207–C218.

Jason S Sobel, Andrew S Forsberg, David H Laidlaw, Robert C Zeleznik, Daniel F Keefe, Igor Pivkin, George E Karniadakis, Peter Richardson, and Sharon Swartz. 2004. Particle flurries. *IEEE Computer Graphics and Applications* 24, 2 (2004), 76–85.

Richard Stoakley, Matthew J Conway, and Randy Pausch. 1995. Virtual reality on a WIM: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co., 265–272.

Marc Swackhamer, Andrea J Johnson, Daniel Keefe, Seth Johnson, Ross Altheimer, and Aaron Wittkamper. 2017. Weather report: Structuring data experience in the built environment. *Proceedings of Architectural Research Centers Consortium* (2017), 102–111.

Natalya Tatarchuk. 2005. Practical dynamic parallax occlusion mapping.. In *SIGGRAPH Sketches*. Citeseer, 106.

Laura G Tateosian, Christopher G Healey, and James T Enns. 2007. Engaging viewers through nonphotorealistic visualizations. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*. ACM, 93–102.

Alex S Taylor, Siân Lindley, Tim Regan, David Sweeney, Vasillis Vlachokyriakos, Lillie Grainger, and Jessica Lingel. 2015. Data-in-place: Thinking through the relations between data and community. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2863–2872.

Annie Thomson. 2019. Design review in VR: The power of immersive collaboration. `https://resources.unity.com/webinars/design-review-in-vr-the-power-of-immersive-collaboration`

Barbara Tversky, Julie Bauer Morrison, and Mireille Betrancourt. 2002. Animation: can it facilitate? *International journal of human-computer studies* 57, 4 (2002), 247–262.

Andries Van Dam, Andrew S Forsberg, David H Laidlaw, Joseph J LaViola, and Rosemary M Simpson. 2000. Immersive VR for scientific visualization: A progress report. *IEEE Computer Graphics and Applications* 20, 6 (2000), 26–52.

virtualdesktop. 2019. Virtual Desktop. `https://www.vrdesktop.net/`.

Dany Vohl, David G Barnes, Christopher J Fluke, Govinda Poudel, Nellie Georgiou-Karistianis, Amr H Hassan, Yuri Benovitski, Tsz Ho Wong, Owen L Kaluza, Toan D

Nguyen, et al. 2016. Large-scale comparative visualisation of sets of multidimensional data. *PeerJ Computer Science* 2 (2016), e88.

Eileen Vote, Daniel Acevedo, Cullen Jackson, Jason Sobel, and David H. Laidlaw. 2003. Design-by-Example: A Schema for Designing Visualizations Using Examples from Art. In *Proceedings of ACM SIGGRAPH 2003 Sketches and Applications*. ACM.

S Wang, D Bailey, K Lindsay, K Moore, and M Holland. 2014. Impacts of sea ice on the marine iron cycle and phytoplankton productivity. *Biogeosciences Discussions* 11 (2014), 2383–2418.

Shanlin Wang, Scott Elliott, Mathew Maltrud, and Philip Cameron-Smith. 2015. Influence of explicit Phaeocystis parameterizations on the global distribution of marine dimethyl sulfide. *Journal of Geophysical Research: Biogeosciences* 120, 11 (2015), 2158–2177.

Matthew O. Ward. 2002. A Taxonomy of Glyph Placement Strategies for Multidimensional Data Visualization. *Information Visualization* 1, 3/4 (2002), 194–210.

Colin Ware. 2012. *Information Visualization: Perception for Design* (3rd ed.). Morgan Kaufman, San Francisco, CA.

Colin Ware and William Knight. 1995. Using visual texture for information display. *ACM Transactions on Graphics (TOG)* 14, 1 (1995), 3–20.

Colin Ware and Peter Mitchell. 2005. Reevaluating stereo and motion cues for visualizing graphs in three dimensions. In *Proceedings of the 2nd symposium on Applied perception in graphics and visualization*. ACM, 51–58.

Colin Ware and Peter Mitchell. 2008. Visualizing graphs in three dimensions. *ACM Transactions on Applied Perception (TAP)* 5, 1 (2008), 2.

Colin Ware, William Wright, and Nicholas J. Pioch. 2013. Visual Thinking Design Patterns. *Distributed Multimedia Systems Proceedings* (2013), 150–155.

Jurgen Waser, Raphael Fuchs, Hrvoje Ribicic, Benjamin Schindler, Gunther Bloschl, and Eduard Groller. 2010. World lines. *IEEE transactions on visualization and computer graphics* 16, 6 (2010), 1458–1467.

Jürgen Waser, Artem Konev, Bernhard Sadransky, Zsolt Horváth, H Ribičić, Robert Carnecky, P Kluding, and Benjamin Schindler. 2014. Many plans: Multidimensional ensembles for visual decision support in flood management. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 281–290.

Brandon K. Wiggins, Francesca Samsel, Kristin Hoch, Greg Abram, Joseph Smidt, Sam Jones, Alex Gagliano, and Morgan Taylor. 2019. The First Water in the Universe. *Proceedings of the 2019 ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis (SC).* Scientific Visualization & Data Analytics Showcase.

Brandon K Wiggins and Joseph Smidt. 2018. Cosmological Simulations with Molecular Astrochemistry: Water in the Early Universe. In *American Astronomical Society Meeting Abstracts*, Vol. 231.

Georges Winkenbach and David H Salesin. 1994. Computer-generated pen-and-ink illustration. In *Proceedings of the 21st annual conference on Computer Graphics and Interactive Techniques*. ACM, 91–100.

Phillip J. Wolfram, Todd D. Ringler, Mathew E. Maltrud, Douglas W. Jacobsen, and Mark R. Petersen. 2015. Diagnosing isopycnal diffusivity in an eddying, idealized mid-latitude ocean basin via Lagrangian In-situ, Global, High-performance particle Tracking (LIGHT). *Journal of Physical Oceanography* 45, 8 (2015), 2114–2133.

Robert C Zeleznik, Joseph J LaViola, D Acevedo Feliz, and Daniel F Keefe. 2002. Pop through button devices for VE navigation and interaction. In *Virtual Reality, 2002. Proceedings. IEEE*. IEEE, 127–134.

Song Zhang, Mark E. Bastin, David H. Laidlaw, Saurabh Sinha, Paul A. Armitage, and Thomas S. Deisboeck. 2004. Visualization and analysis of white matter structural asymmetry in diffusion tensor MRI data. *Magnetic Resonance in Medicine* 51, 1 (2004), 140–147.

Kun Zhao, Naohisa Sakamoto, and Koji Koyamada. 2017. Using interactive particle-based rendering to visualize a large-scale time-varying unstructured volume with mixed cell types. In *Pacific Visualization Symposium (PacificVis), 2017 IEEE*. IEEE, 185–189.

L. Zhou and C. D. Hansen. 2016. A Survey of Colormaps in Visualization. *IEEE Transactions on Visualization and Computer Graphics* 22, 8 (Aug. 2016), 2051–2069. `https://doi.org/10.1109/TVCG.2015.2489649`