ABSTRACT

Title of dissertation: SALIENCY-GUIDED
GRAPHICS AND VISUALIZATION

Youngmin Kim, Doctor of Philosophy, 2008

Dissertation directed by: Professor Amitabh Varshney
Department of Computer Science

In this dissertation, we show how we can use principles of saliency to enhance depiction, manage visual attention, and increase interactivity for 3D graphics and visualization. Current mesh saliency approaches are inspired by low-level human visual cues, but have not yet been validated. Our eye-tracking-based user study shows that the current computational model of mesh saliency can well approximate human eye movements. Artists, illustrators, photographers, and cinematographers have long used the principles of contrast and composition to guide visual attention. We present a visual-saliency-based operator to draw visual attention to selected regions of interest. We have observed that it is more successful at eliciting viewer attention than the traditional Gaussian enhancement operator for visualizing both volume datasets and 3D meshes.

Mesh saliency can be measured in various ways. The previous model of saliency computes saliency by identifying the uniqueness of curvature. Another way to identify uniqueness is to look for non-repeating structure in the middle of repeating structure. We have developed a system to detect repeating patterns in 3D point datasets. We introduce

the idea of creating vertex and transformation streams that represent large point datasets via their interaction. This dramatically improves arithmetic intensity and addresses the input geometry bandwidth bottleneck for interactive 3D graphics applications.

Fast-previewing of time-varing datasets is important for the purpose of summarization and abstraction. We compute the salient frames in molecular dynamics simulations through the subspace analysis of the protein's residue orientations. We first compute an affinity matrix for each frame i of the simulation based on the similarity of the orientation of the protein's backbone residues. Eigenanalysis of the affinity matrix gives us the subspace that best represents the conformation of the current frame i. We use this subspace to represent the frames ahead and behind frame i. The more accurately we can use the subspace of frame i to represent its neighbors, the less salient it is.

Taken together, the tools and techniques developed in this dissertation are likely to provide the building blocks for the next generation visual analysis, reasoning, and discovery environments.

SALIENCY-GUIDED GRAPHICS AND VISUALIZATION

by

Youngmin Kim

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Advisory Committee:
Professor Amitabh Varshney, Chair/Advisor
Professor David W. Jacobs
Professor Joseph F. JaJa
Professor David Mount
Professor Sung Lee

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible. First and foremost I would like to express my sincere gratitude to my advisor, Amitabh Varshney for the freedom and the encouragement to pursue challenging and extremely interesting projects that led to this dissertation. He has always made himself available for advice, and helped me develop the best characteristics for my research. It has been my great pleasure to work with and learn from such an extraordinary person. I would also like to thank David Jacobs. Without his invaluable theoretical ideas and insights in human visual perception, this thesis would have been a distant dream. I thank Joseph JaJa, David Mount, Sung Lee, Sergei Sukharev, Andriy Anishkin, Dianne P. O'Leary, and François Guimbretière for their excellent suggestions for improving this dissertation.

I would like to give my special thanks to the entire members of the graphics and visual informatics laboratory, who have been a joy to work with over the years. I thank all other friends in the University of Maryland for their help and encouragement.

I wish to thank my family - my parents, sister, and brother who have always supported me throughout my life. Finally, my deepest appreciation goes to my wife for her love and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The last decade has seen a phenomenal growth in the complexity of 3D graphics and scientific visualization models due to the advances in acquisition and simulation technologies. This trend coupled with the recent advances in graphics hardware and displays has brought us to a stage where we now have an unprecedented ability to model and render detail. The human visual system, though impressive and complex, has its limits in comprehending detail. An important issue here is how to exploit this largely fixed bandwidth channel of the human visual system to allow users to comprehend information more quickly. In this dissertation, we incorporate principles of saliency inspired by human perception into the traditional graphics and visualization pipeline to augment analysis and image synthesis for 3D datasets.

As a first step towards our goal, we would like to have a model of saliency that has been validated with respect to human performance. Even though the previously proposed model of mesh saliency [69] uses a center-surround mechanism which is inspired by the human visual system, it has not yet been validated by comparing it with human eye movements. We examine the previous model of mesh saliency by quantifying the similarity between the model and human eye fixations.

Once we have a validated model of saliency, we discuss how to draw viewers atten-

tion to desired regions of interest by inverting this computational model of saliency. The guidance of attention can be helpful for facilitating communication and exploration of a 3D dataset. A content creator or a domain expert can identify high importance regions and use this technique in helping viewers navigate and understand complex datasets. We can also apply this approach to exploratory visualization systems that rely on automated and fuzzy detection of features. Such systems could use saliency-based perceptual enhancement to generate a variable level of perceptual interest to the human observer.

Designing a high-level representation to extract important components of the dataset is useful not only for analyzing but also for improving the interactivity of graphics and visualization applications. For example, looking for the most salient repeating structure can help in succinctly representing a 3D dataset as well as communicating it efficiently between the CPU and the GPU. As the gap between processing speeds and memory access times grows ever wider, the impact of this kind of high-level representation on graphics rendering performance should rise even further.

Saliency-based analysis can be applied to time-varying 3D datasets for the purpose of summarization, abstraction, and motion analysis. As the sizes of time-varying datasets continue to grow, it becomes more and more difficult to comprehend them. Automatically generated thumbnail images and previewing of time-varying datasets can help viewers explore and understand the datasets significantly faster as well as provide new insights.

1.2 Dissertation Hypothesis

My dissertation hypothesis is that principles of saliency inspired by the human

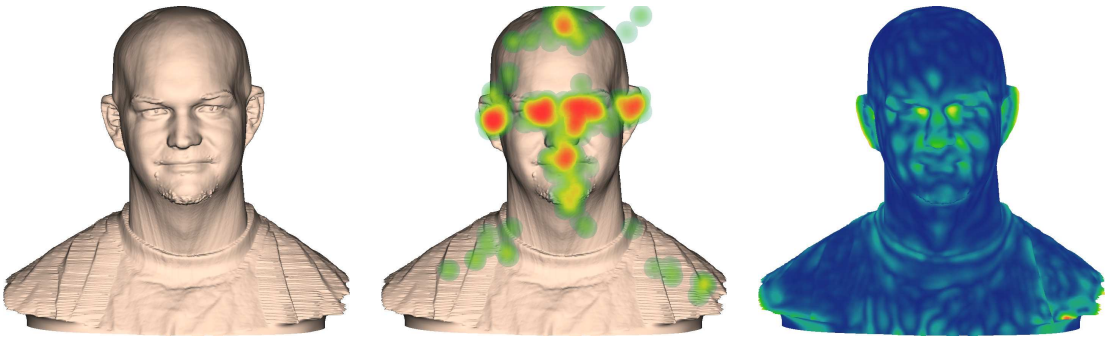
visual system are important in enhancing depiction, managing visual attention, and increasing interactivity for 3D graphics and visualization.

I have designed and implemented the following systems to address my research hypothesis:

- *Saliency Validation:* We examine how well the existing model of saliency correlates with the human visual system through an eye-tracking-based user study. Having a validated model of mesh saliency is greatly helpful for identifying the role of 3D information in the human visual system as well as building further saliency-based systems for tasks such as visual enhancement.
- *Visual Enhancement:* We present a novel visual-saliency-based operator to enhance selected regions of interest both in volume datasets and in 3D meshes. We have found that it is more successful at eliciting viewer attention than the traditional Gaussian regional enhancement approach.
- *Salient Transformation Streams:* We introduce the idea of creating vertex and transformation streams that represent large point data sets via their interaction. We identify the most salient repetitive patterns in 3D point datasets, and use that information to improve the rendering rate.
- *Salient Frame Detection:* We introduce a salient frame detection technique based on motion subspaces analysis for large time-varying datasets. We validate our approach for molecular dynamics simulations.

1.3 Saliency Validation

1.3.1 Problem Definition



Original Human Eye Fixations Mesh Saliency

Figure 1.1: *Human eye fixations and Mesh Saliency on the Cyberware Male model. Warm colors in the saliency map indicate high saliency regions while cool colors indicate low saliency regions.*

With the increase in size, number, and complexity of 3D graphics datasets, it will become increasingly important to integrate principles of saliency with geometric processing of meshes. Lee *et al.* [69] have proposed a model of mesh saliency as a measure of regional importance. Their method for computing mesh saliency uses a center-surround mechanism that is inspired by the human visual system. Similar mechanisms have been widely used in models of 2D image saliency [52] [66], and their models were validated with respect to human performance [91] [99]. As far as we know, there has been no work in comparing models of 3D saliency to eye movements, although many experiments have measured eye movements as participants examine 3D objects [18] [49] [74].

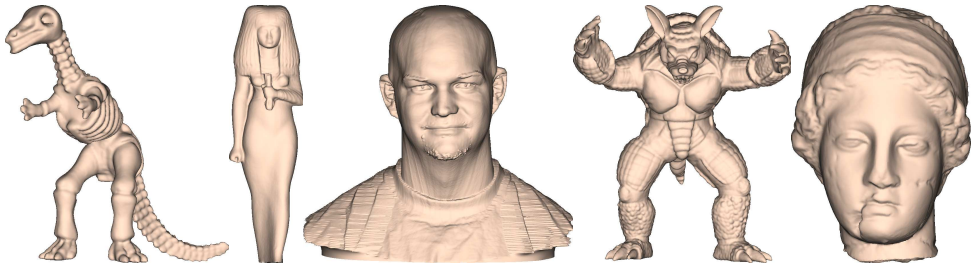
The problem we wish to address through user studies is how effective the model of mesh saliency as proposed by Lee *et al.* [69] is in modeling human eye movements. Having a validated model of mesh saliency will be likely useful in several contexts. For

example, it could be helpful for identifying the role of 3D information in a visual search task as Enns and Rensink have explored in their work [30]. It can be also helpful for designing a better visual saliency model which is closer to human eye movements.

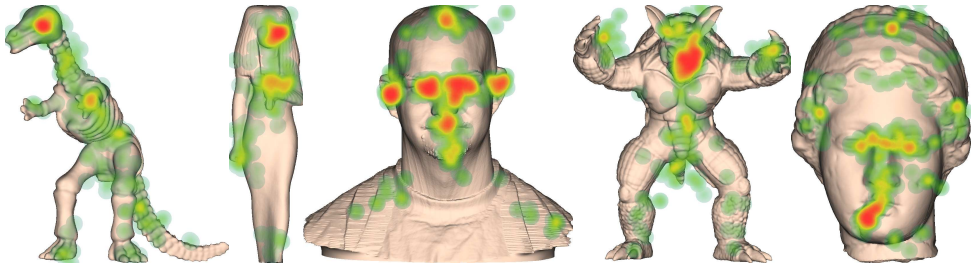
1.3.2 Our Approach

In Chapter 2 we present a user study that compares the previous mesh saliency approaches with human eye fixations. We quantify how much better a computational model of saliency models human eye movements than can be expected by chance or by a curvature-based model for 3D rendered images. The main contributions of this dissertation on saliency validation are:

1. We compare models of 3D mesh saliency to eye movements through an eye-tracking-based user study.
2. We introduce the normalized chance-adjusted saliency to quantify the correlation between mesh saliency and fixations for 3D rendered scenes. This is more appropriate for 3D rendered scenes than the previous chance-adjusted saliency measure which was designed for 2D images.
3. We show that the current computational model of mesh saliency models human eye movements significantly better than can be expected by chance or due to curvature alone.



(a) Original Models



(b) Human Eye Fixations

Figure 1.2: *The Dinosaur, Isis, Male, Armadillo, and Igea models and recorded Human Eye Fixations in our study. Here, we represent human eye fixations with hot-spot maps where warm colors indicate highly-fixated regions.*

1.3.3 Results of Saliency Validation

We have used five natural scanned models for our user study. Figure 1.2 illustrates these models and human eye fixations recorded for the first 5 seconds from 18 participants. Fixation points are illustrated with hot-spot maps, where warm colors show highly-fixated regions.

Figure 1.1 illustrates the fixation points and computed mesh saliency for the Male model, and Figure 1.3 further illustrates the fixation points, computed mean curvature, and mesh saliency for the Dinosaur model. The curvature-based model has been used for comparison with the mesh saliency model. We observe that most fixation points are closely related to the mesh saliency model.

To quantify how each of saliency models (curvature-based and mesh saliency) cor-

(a) Eye Fixations (b) Mean Curvature (c) Mesh Saliency

Figure 1.3: *The saliency models and human eye fixations on the Dinosaur model in Figure 1.2.*

relates to human eye fixations, we have introduced normalized chance-adjusted saliency. This measurement will be discussed in great details in Chapter 2. We note that the bigger the normalized chance-adjusted value is, the better correlation each saliency model has with human eye fixations. The normalized chance-adjusted saliency allows us to quantify how much better a computational model of saliency models human eye movements than can be expected by chance for 3D rendered images. It can be also used to indirectly compare which saliency models (curvature-based and mesh saliency) better correlate with human eye fixations.

Figure 1.4 shows the average normalized chance-adjusted saliency values computed by the curvature model and the mesh saliency model across all participants. In general, we have observed that both computational models of saliency have higher correlation with human eye fixations than a random model as the normalized chance-adjusted saliency values are higher than 1, the value that can be expected purely by chance. We observe lower variances in normalized chance-adjusted saliency than chance-adjusted saliency cases.

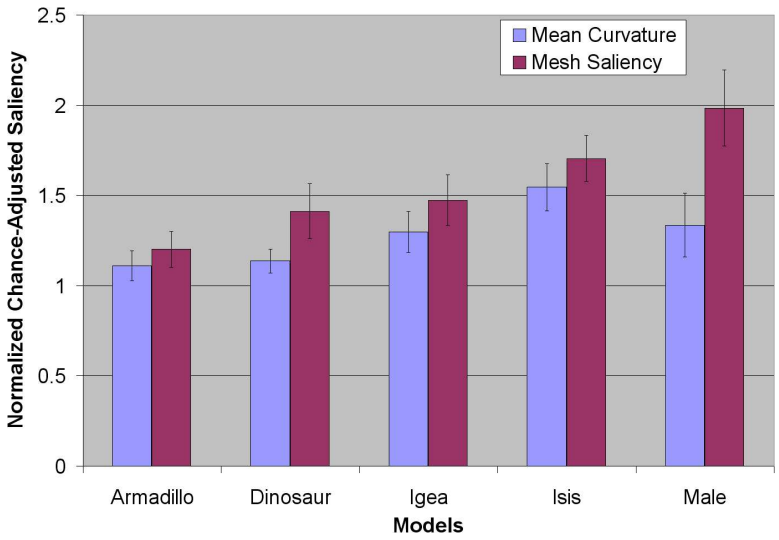


Figure 1.4: *Average Normalized Chance-Adjusted Saliency Values and 95% Confidence Interval using Curvature and Mesh Saliency across All Participants for each Model. For all the cases, the values are higher than 1, which is the value that can be expected by chance.*

Our analysis shows that the mesh saliency model has a significantly higher correlation with human eye fixations than a random model and a curvature-based model.

1.4 Visual Enhancement for 3D Meshes

1.4.1 Problem Definition

Visual attention can be guided in a goal-driven fashion or by external stimuli [98]. In this work we focus on the latter and therefore do not consider the high-level semantics of the objects or tasks at hand. Visual attention can be drawn to a specific region by simply having the selected pixels rapidly change and flash colors. Other approaches to draw attention to a region could include lighting it brightly, using high-saturation colors, or adding a high-curvature spike. Although these approaches would likely work, they should be considered coercive and obtrusive instead of being persuasive. The challenge

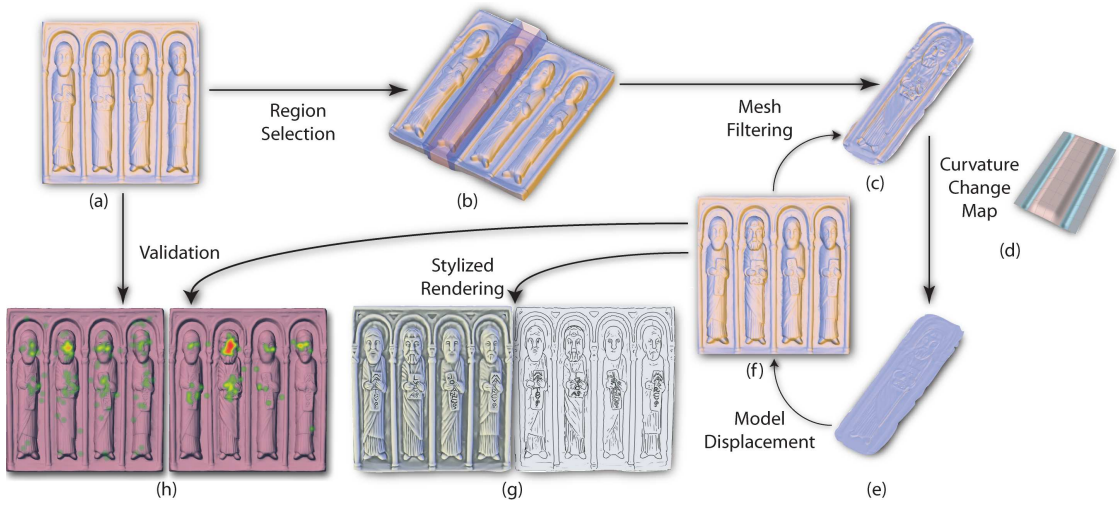


Figure 1.5: *An overview of persuading visual attention through geometry. The content creator defines a region of attention over the mesh in (a) as shown in (b); Mesh filtering over the desired attention region provides a set of displacements (c) along the vertex normals; Vertex displacements are weighted by a curvature change map (d) and then added to the input mesh. The resulting mesh in (f) elicits greater visual attention in the desired region (the second saint). Further, the mesh retains its visual attention persuasiveness through various rendering styles and illuminations (g) and this is validated by eye-tracking-based user study as shown in (h).*

in gently guiding visual attention is to do so in a finely nuanced style that only introduces subtle changes. We have examined how saliency alteration in 3D geometry may affect viewer attention.

1.4.2 Our Approach

In Chapter 3 we explore visual attention persuasion by making changes only to geometry. The advantage of pursuing visual attention persuasion through geometry manipulation is that by pushing the influence of attention deeper into the graphics pipeline, content creators can have greater flexibility in using other conventional techniques later in the graphics pipeline. The various stages in our approach are shown in Figure 1.5 and summarized below.

1. We allow the content designer to specify attentional regions in a scene by directly selecting vertices, regions, or objects.
2. We discuss an approach to enhancing the saliency of a region to attract greater visual attention by designing a general class of mesh filters.
3. Once a persuasion filter has been applied to a given region we would like to have some objective evidence that it results in eliciting greater visual attention. We have conducted an eye-tracking-based user study to verify the impact of our persuasion filters.
4. Our approach of geometry filtering incorporates visual attention persuasion early in the graphics pipeline. We have empirically observed that these changes are successfully propagated to the final rendering under several illumination and rendering styles.

1.4.3 Results of Visual Enhancement for Meshes

We have empirically tested our models with and without the application of the persuasion filters with several illumination and rendering styles. These include the standard OpenGL lighting model and suggestive contours [23]. The results of the application of our persuasion filters are clearly discernible with each of these lighting models and illumination styles. These are shown in Figure 1.6. The reason for the successful propagation of fine geometry alterations to the final rendered image could be that the bilateral displacements allow us to preserve and enhance edges in the target attention area, while smoothing them around it. These effects are perhaps most clearly visible in the suggestive contour

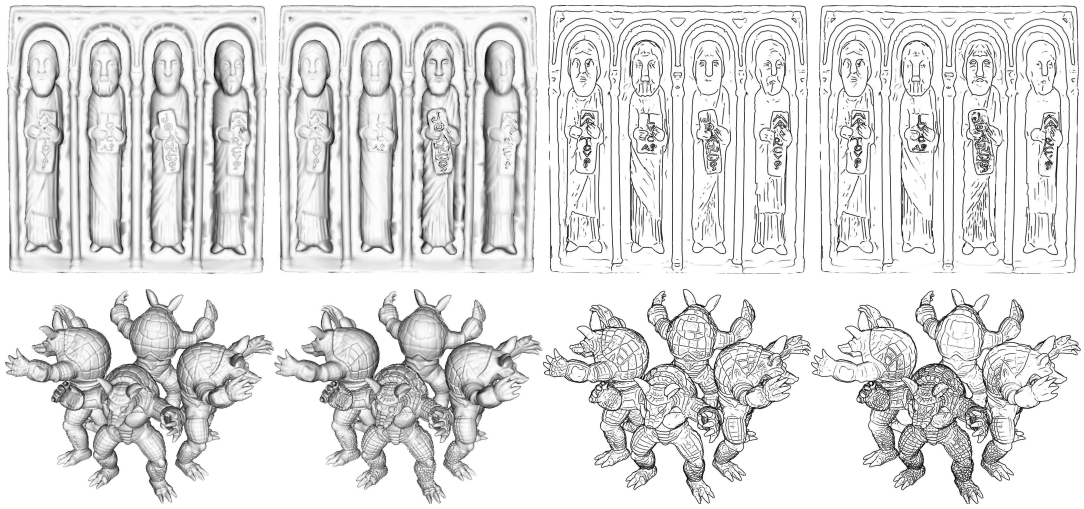


Figure 1.6: *Illustration of the Romanesque Relief and the Armadillos by Lambertian lighting and Suggestive Contours. The first and the third column show the original models while the second and the fourth column have persuasion filters applied to the third saint with* $(\lambda_+ = 0.2, \lambda_- = 0.1)$ *and to the frontmost Armadillo with* $(\lambda_+ = 0.3, \lambda_- = 0.3)$*, respectively.*

rendering emphasizing the third saint and the closest Armadillo in Figure 1.6.

To gather objective evidence of the effectiveness of our approach we have carried out an eye-tracking-based user study and have analyzed the results on the percentage of fixation points on the region of interest. We compare our results with prior art (the Gaussian-based enhancement method). The Gaussian fall-off function has been used in regional enhancement for volume illustration [104], in attention-based modulation of detail for exaggerated shading [106], and in stylized rendering [18].

The results of our user study can be seen in Figure 1.7 and they show the increase in fixation points on the regions selected by the user. Each grouping of bars in Figure 1.7 is labeled by the object or region on which the filters was applied. Figure 1.8 shows the increase in the number of fixation points on the region of interest after the application of persuasion filters. In Figure 1.8(b), the third saint was processed with persuasion filters

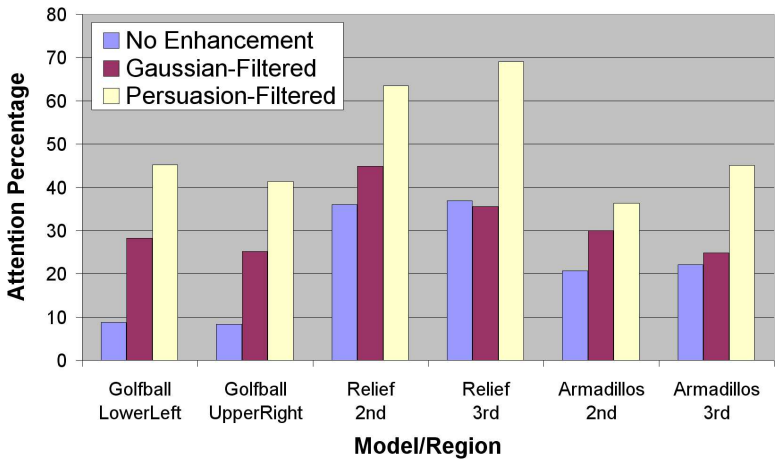
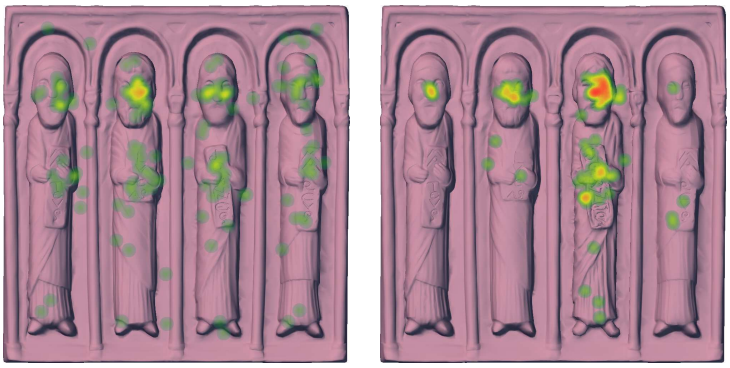


Figure 1.7: *The bars show the average percentage of fixation points on the region of interest for the original, the Gaussian-filtered and the Persuasion-filtered models.*



(a) (b)

Figure 1.8: *Image(a) and (b) show the fixation points on the original model and the model altered by persuasion filter, respectively. Fixation points are recorded over the first* 5 *seconds from* 9 *subjects, and visualized with hot spot map where warm colors show the areas of highest fixation count.*

and this resulted in an 87% increase in the percentage of fixation points on it.

We have carried out a two-way ANOVA on the percentage of fixations for two conditions: filtering methods and image sets. As shown in Figure 1.7, participants fixated more on the regions of interest when they are filtered with enhancement techniques. Overall, there was a strong difference for the percentage of fixations depending on enhancement techniques, $F(2,153) = 30.09, p < 0.001$.

Since we filtered two regions of interest for each image set, we did another analysis

Table 1.1: *The ANOVA Tests*

Condition	F-Value	p-Value
Enhancement techniques on three sets	30.09	< 0.001
Enhancement techniques on Golfball	13.45	< 0.001
Enhancement techniques on Relief	9.728	< 0.001
Enhancement techniques on Armadillos	7.2	0.002
Gaussian vs. Persuasion on three sets	21.23	< 0.001

Table 1.2: *The Pairwise t-tests (two-tailed)*

Model	Condition	t-Value	p-Value
Golfball	No Change vs. Gaussian	-3.34	0.004
	No Change vs. Persuasion	-5.84	< 0.001
	Gaussian vs. Persuasion	-2.81	0.012
Relief	No Change vs. Gaussian	-0.47	0.647
	No Change vs. Persuasion	-4.69	< 0.001
	Gaussian vs. Persuasion	-3.95	0.001
Armadillos	No Change vs. Gaussian	-1.56	0.138
	No Change vs. Persuasion	-3.42	0.003
	Gaussian vs. Persuasion	-2.31	0.033

with filtering methods and regions for each image set. Amongst regions, there were no statistically significant differences ($F(2,48) = 0.027 \sim 0.195, p > 0.661$) as expected. For enhancement methods, there were statistically significant differences for all the models. We also carried out a two-way ANOVA test with two enhancement methods (Gaussian and Persuasion) to see if the Persuasion-based enhancement is better. The result shows a significant improvement on the percentage of fixations with Persuasion-based enhancement, $F(1,102) = 21.23, p < 0.001$.

The results in this section clearly validate a significant increase of fixations on the regions of interest by the persuasion filter over the original as well as the Gaussian-filtered meshes.

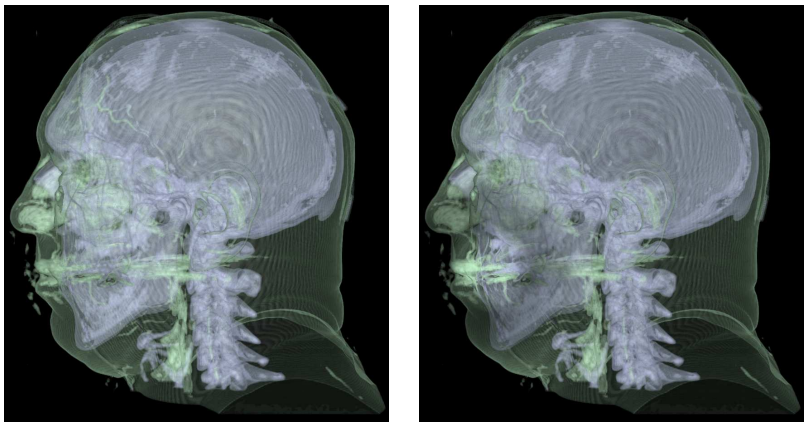
1.5 Visual Enhancement for 3D Volume Datasets

1.5.1 Problem Definition

Visually depicting large volume datasets in a comprehensible way has been a long-standing challenge. Transfer functions have been widely used to help visualize the features and details in volumes by assigning varying optical properties such as color and opacity to different densities of a volumetric scalar field. Significant advances have been made in the art and the science of devising transfer functions that successfully show the inherent structures within a given volume dataset. Despite these impressive advances the transfer functions remain a mapping of the physical appearance to the local geometric attributes such as the local density of the scalar field and its first and higher-order derivatives. Notwithstanding the pioneering work in dual-domain interactions by Kniss *et al.* [64], transfer functions by and large remain ill-suited to directly afford the appearance manipulation of selected regions of a volume. As the volume datasets have grown in complexity, so too has the need to emphasize and draw visual attention to appropriate regions in their visualization.

1.5.2 Our Approach

In Chapter 4 we address the growing need for tools and techniques that can draw visual attention to user-specified regions in a direct volume-rendering environment. Towards this goal we seek solutions based on multi-scale methods for visual saliency that can be used to guide visual attention based on varying perceptual importance.



(a) (b)

Figure 1.9: *Saliency-guided Enhancement for Volume Visualization: Image (a) shows the traditional volume visualization and image (b) shows the result of applying our saliency-guided enhancement operator to the mouth.*

We have introduced a new visualization enhancement operator that is inspired by the center-surround mechanism of visual saliency. Our goal is to enhance human perception of the volume data by guiding a viewer's attention to specific regions of interest. Since our method considers the influence of each voxel at multiple scales, it can emphasize volumetric features at an appropriate visual scale. Existing transfer functions, based on local geometry and its derivatives, would find it difficult to achieve a similar level of multi-scale emphasis. Our saliency-guided enhancement framework provides scientists and medical researchers a valuable tool to enable them to easily emphasize and de-emphasize regions of interests even in large volume datasets, successfully guiding user's visual attention to desired regions without sacrificing their local context. Saliency-guided emphasis is likely to find use in large-scale visual knowledge discovery applications where knowledge discovery modules could identify the regions satisfying a certain criteria and then present them visually with subtle variations to draw a user's attention to those regions in order of their importance.

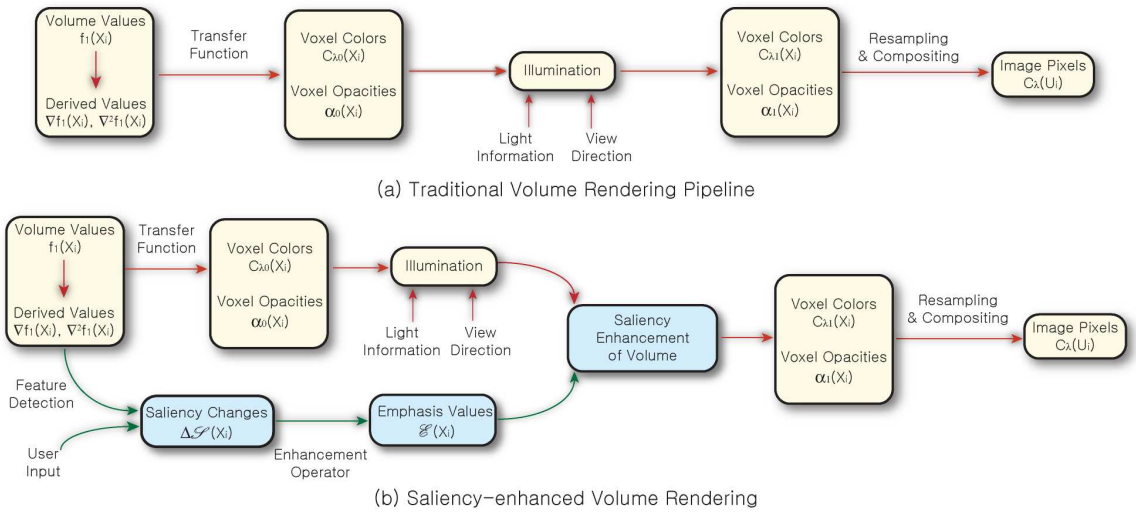


Figure 1.10: *(a) The traditional visualization pipeline. (b) Saliency-enhanced visualization pipeline. The saliency field is modified by the enhancement operator to generate an emphasis field. The emphasis field is used to enhance the perception of features in volume by modulating appearance attributes such as luminance, chrominance, and texture detail.*

The main contributions of this dissertation on visual enhancement for 3D volume datasets are:

1. We present a new saliency-based enhancement operator to guide visual attention in volume visualization.
2. We discuss augmenting the existing visualization pipeline by incorporating enhancement operators to increase the visual saliency of different regions of a volume dataset.
3. We present an eye-tracking-based user study that shows that our saliency-enhancement operator is successful in eliciting viewer attention in volume visualization.

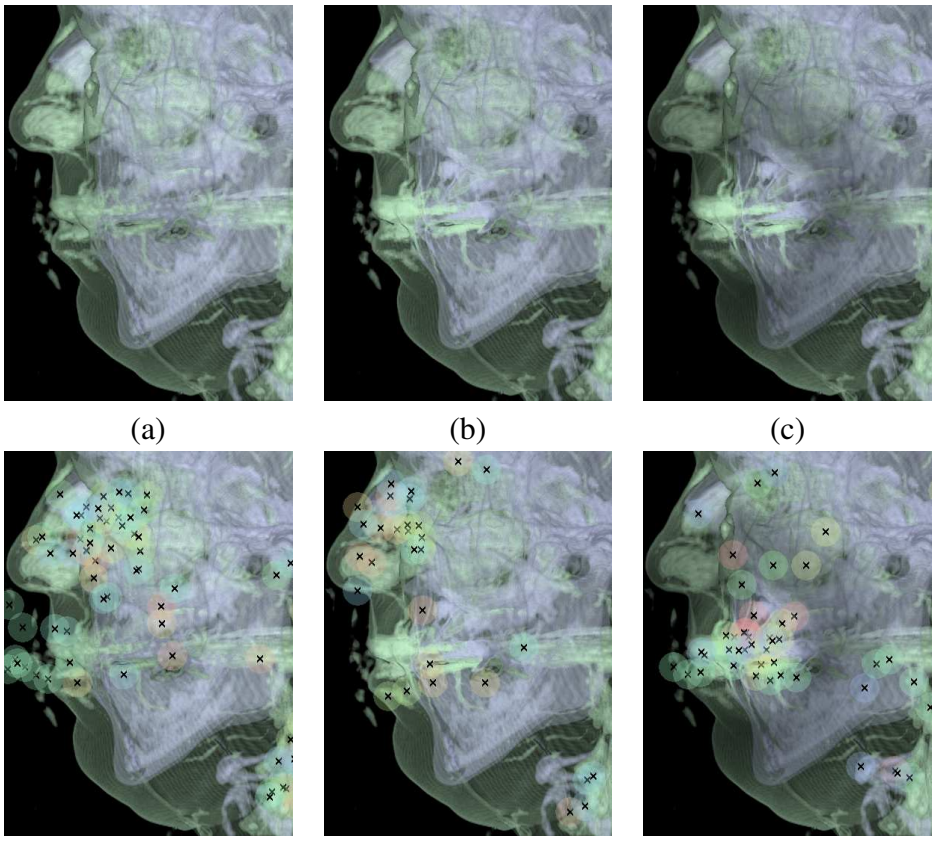


Figure 1.11: *The Visible Male model (*$128 \times 256 \times 256$*) is rendered by the traditional volume visualization in image (a). Images (b) and (c) show the visualization with regional enhancement by a Gaussian and our saliency-guided operator, respectively. Images (d), (e), and (f) show the fixation points collected from our user study with the images (a), (b), and (c), respectively.*

1.5.3 Results of Visual Enhancement for 3D Volume Datasets

We show our results by changing the value parameter in the HSV color model. Figure 1.11 compares the enhancement by a traditional Gaussian operator and by our new saliency-guided enhancement operator on the Visible Male model. Notice that the original image has high brightness regions such as the nose. While the Gaussian operator increases the brightness of the user-specified regions, our saliency-enhancement operator additionally lowers the brightness in the neighborhood. This difference results in a

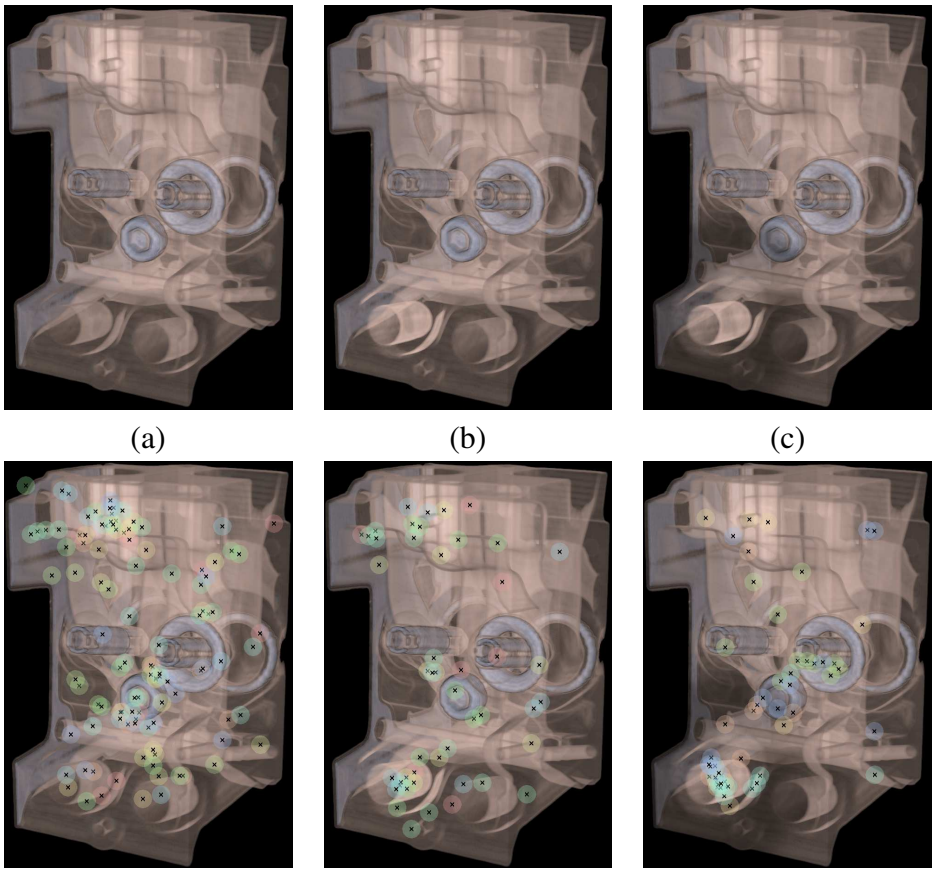


Figure 1.12: *The Engine Block model (*$256 \times 256 \times 256$*) is rendered by the traditional volume visualization in image (a). Images (b) and (c) show the visualization with regional enhancement by a Gaussian and our saliency-guided operator, respectively. Images (d), (e), and (f) show the fixation points collected from our user study with the images (a), (b), and (c), respectively.*

much greater user attention to the desired regions, even with subtle changes to the overall brightness. Figure 1.12 shows another comparison on the Engine Block model.

We have carried out an eye-tracking-based user study to gather objective evidence of the effectiveness of our approach. Our goal in this user study is to validate our ability to draw a viewer's attention by subtle changes to the appearance of the volume data. We have used the volume datasets of the Engine Block, the Foot, the Visible Male, and the Sheep Heart model for our study. The results of our study are shown in Figure 1.13. Each

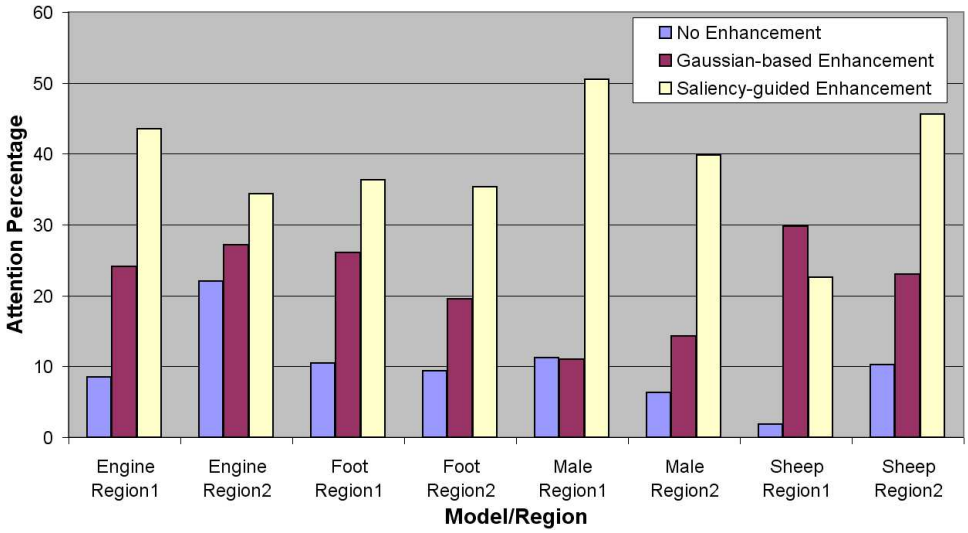


Figure 1.13: *Fixation results for volume visualization enhancements.*

Table 1.3: *List of pairwise t-tests.*

Model	Condition: No Change vs.	t-Value	p-Value
Engine Block	Gaussian-based enhancement	-2.36	0.042
	Saliency-guided enhancement	-2.86	0.019
Foot	Gaussian-based enhancement	-2.67	0.026
	Saliency-guided enhancement	-3.34	0.009
Visible Male	Gaussian-based enhancement	-0.661	0.525
	Saliency-guided enhancement	-6.65	< 0.001
Sheep Heart	Gaussian-based enhancement	-3.86	0.005
	Saliency-guided enhancement	-4.49	0.002

grouping of bars shows the percentage of fixations that fell in a desired region for the unaltered, Gaussian-enhanced, and Saliency-enhanced visualizations for a specific model and region on that model.

We next carried out a pairwise t-test on the percentage of fixations before and after we apply enhancement techniques for each model (this is the only condition in the test). Table 1.3 shows the results from all the models. We found a significant difference in the percentage of fixations when we applied saliency-guided enhancement for all the models.

There was a difference for the percentage of fixations when we applied Gaussian-based enhancement for all the models other than the Visible Male model.

1.6 Salient Transformation Streams

We believe looking for non-repeating structure in the middle of repeating structure is another way to identify the salient regions. Figure 1.14 shows five examples where detecting repeating patterns could be useful for identifying salient parts. Human observers seeing Figure 1.14(a) are likely to characterize the pipe as being more salient than any brick. This could be because the bricks are repeating and the pipe is not. Humans tend to spend more time on salient features that have attracted their attention. This led us

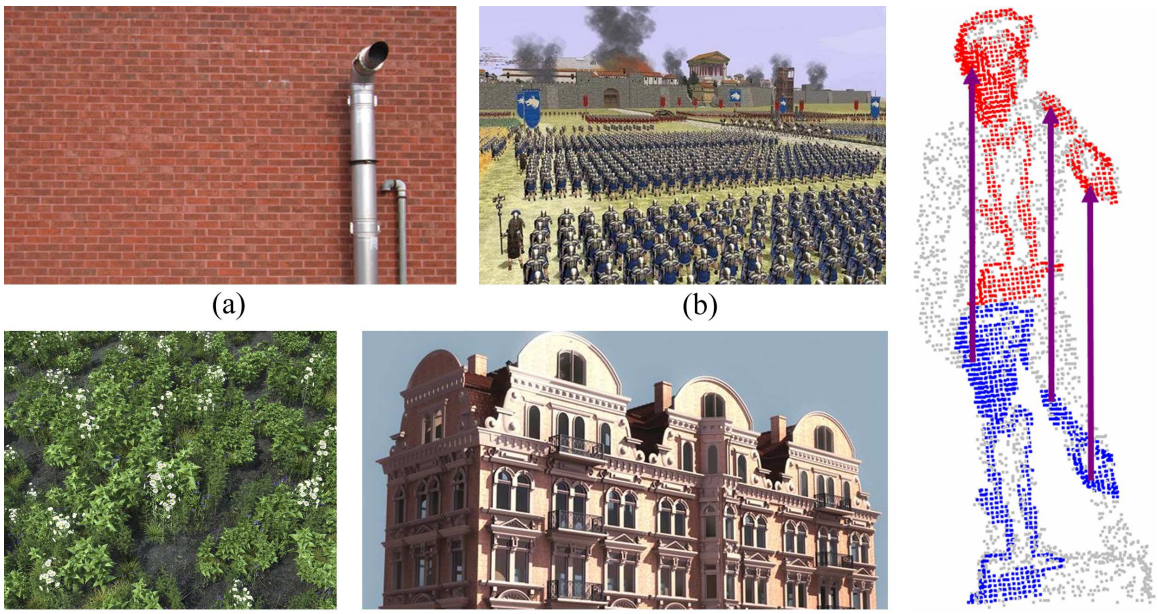


Figure 1.14: *Image (a) shows a pipe against a brick wall at the horitcultural center in Philadelphia's Fairmount Park. Image (b) shows a battlescape generated by Oczone.com, and Image (c) shows an organic model created by the XfrogPlants software. Image (d) shows repeating geometric patterns in the Palace model. Image (e) shows an example of frequently occurring common translations of vertices in Stanford's David model.*

to wonder if allocation of computational resources based on saliency could be used to enhance the efficiency of the graphics pipeline.

Modern graphics processors currently assign the same default importance to every primitive. However, it could be argued that the work done in drawing the first brick could be reused in drawing the second and subsequent bricks due to their similarity. This would suggest a design of the graphics pipeline where the processing was proportional to the saliency of a primitive. Repetitive patterns lend themselves well to succinct representations. The challenge is in automatic discovery of such patterns. It is easy to find such repeating patterns in crowd models (Figure 1.14(b)), in organic models such as trees and flowers (Figure 1.14(c)), or in 3D architectural objects such as the Palace (Figure 1.14(d)).

Even in the Stanford's digital representation of Michelangelo's David model, we can find translational self-similarities of points shown in Figure 1.14(e). This is not as visually salient as the other examples in Figure 1.14. Here, we show how the high-level representation to extract important components of the dataset can be used for improving the interactivity of graphics applications.

1.6.1 Problem Definition

Recent advances in the development of a stream programming environment for GPUs [14] and the use of stream programming for GPUs [43] have enabled graphics researchers and practitioners alike to view graphics operations in the context of data parallel semantics. The stream programming abstraction allows us to consider graphics primitives as streams of records and graphics operations as kernels that operate on such streams. This

simple and yet compelling abstraction has not only had a powerful impact on graphics applications, it has also enabled a wide variety of applications from diverse disciplines such as scientific computing, machine learning, signal processing, computer vision, real-time audio, and computational biology to be mapped on to the GPUs. An important factor behind this success has been the power of the stream programming abstraction to embody, implicitly or explicitly, several important parallel algorithm design considerations such as data parallelism, task parallelism, coherence and latency of memory accesses, producer-consumer locality, and arithmetic intensity.

As the size of a floating-point unit on a 90 nm chip has decreased to almost 0.1% of its area, the challenge has gradually shifted away from trying to accommodate multiple processing units on a single chip towards maximizing the returns from the available bandwidth. In other words, arithmetic is cheap and bandwidth is the critical problem [21].

1.6.2 Our Approach

In Chapter 5 we present a novel method to dramatically enhance the arithmetic intensity (the compute to bandwidth ratio) [22], for vertex streams on the GPUs. The inspiration for our work lies in the idea that two interacting streams are significantly more powerful than a single stream. This basic idea has been around in computer architecture for a while (it was used in IBM 7950 as early as 1961 [8]) but its applications to graphics were not yet possible due to lack of programmable hardware support at the graphics processor level. The recent emergence of instance streams in modern GPUs [28] has allowed us to formulate and validate our ideas on representing geometry as two interacting

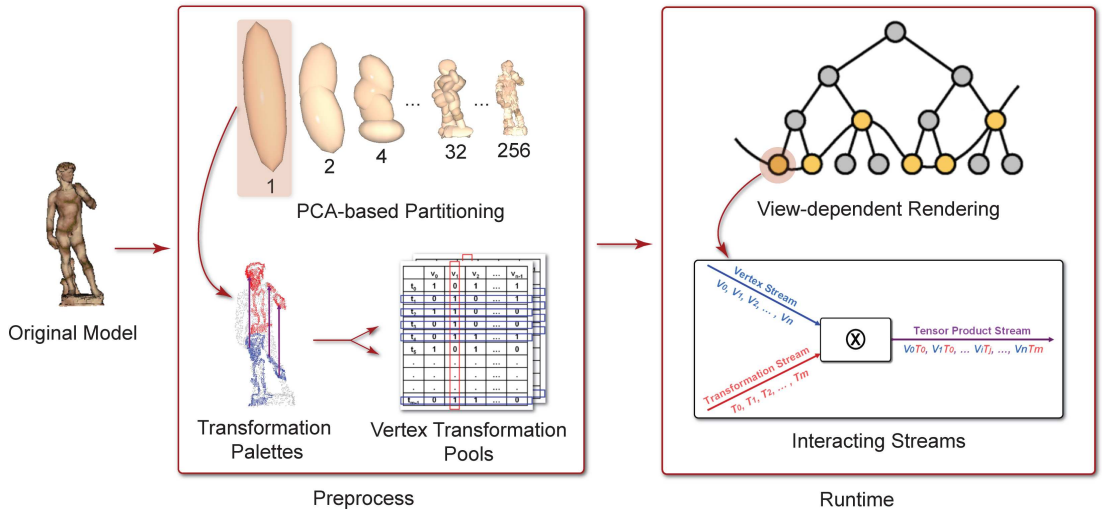


Figure 1.15: *Overview of Salient Transformation Streams. As a preprocess, the PCA analysis of a group of points results in a binary partition tree. Each node of the tree is divided into its child nodes depending on the positions at its local orientation frame. In transformation palettes stage, we identify the most common translations among the vertices in each node. Then, we construct two vertex transformation pools by which we cover as many vertices as possible. At runtime, a view-dependent manager determines the cut in the tree and each node of the cut is visualized by the interaction between vertex and transformation streams.*

streams of coordinates and transformations.

The main contributions of this dissertation on salient transformation streams are:

1. We introduce the idea of interacting vertex and transformation streams to encode general point cloud datasets and discuss how these streams can be decoded using modern vertex shaders.
2. We discuss how to efficiently build vertex and transformation streams from a pool of paired vertices and transformations.
3. We outline a method to identify the most common transformations that can map a set of vertices to itself using the Fast Fourier Transform.
4. We show how our approach of using transformation streams can improve the arithmetic intensity in a view-dependent rendering application.

1.6.3 Results of Salient Transformation Streams

We have validated the results of our approach to efficiently identify and use interacting streams of vertex and transformation data on a number of 3D graphics models using a view-dependent point rendering system. We have run our experiments on a 1.6 GHz Pentium IV Windows PC with 2 GB RAM with a NVIDIA GeForce 6800 Ultra AGP graphics card. We have used the geometry instancing hardware feature in vertex shader 3.0 model and used `DrawIndexedPrimitive( )` command in DirectX 9.0 API.

We have compared our results along two dimensions of performance – the improvement in CPU-GPU communication bandwidth and the improvement in the frame rates. The left and the center columns of Figure 1.16 show the conventional rendering and the rendering by Salient Transformation Streams, respectively. The right column of Figure 1.16 shows the rendering of the vertices covered by the vertex-transformation pools. Two most salient vertex transformation pools are identified by our algorithm, and they cover about 80% of all the vertices. The vertices that are not covered by these two pools are rendered without any geometry instancing using conventional rendering. The gains shown include the overhead of sending singleton vertices that our transformation streams model could not cover.

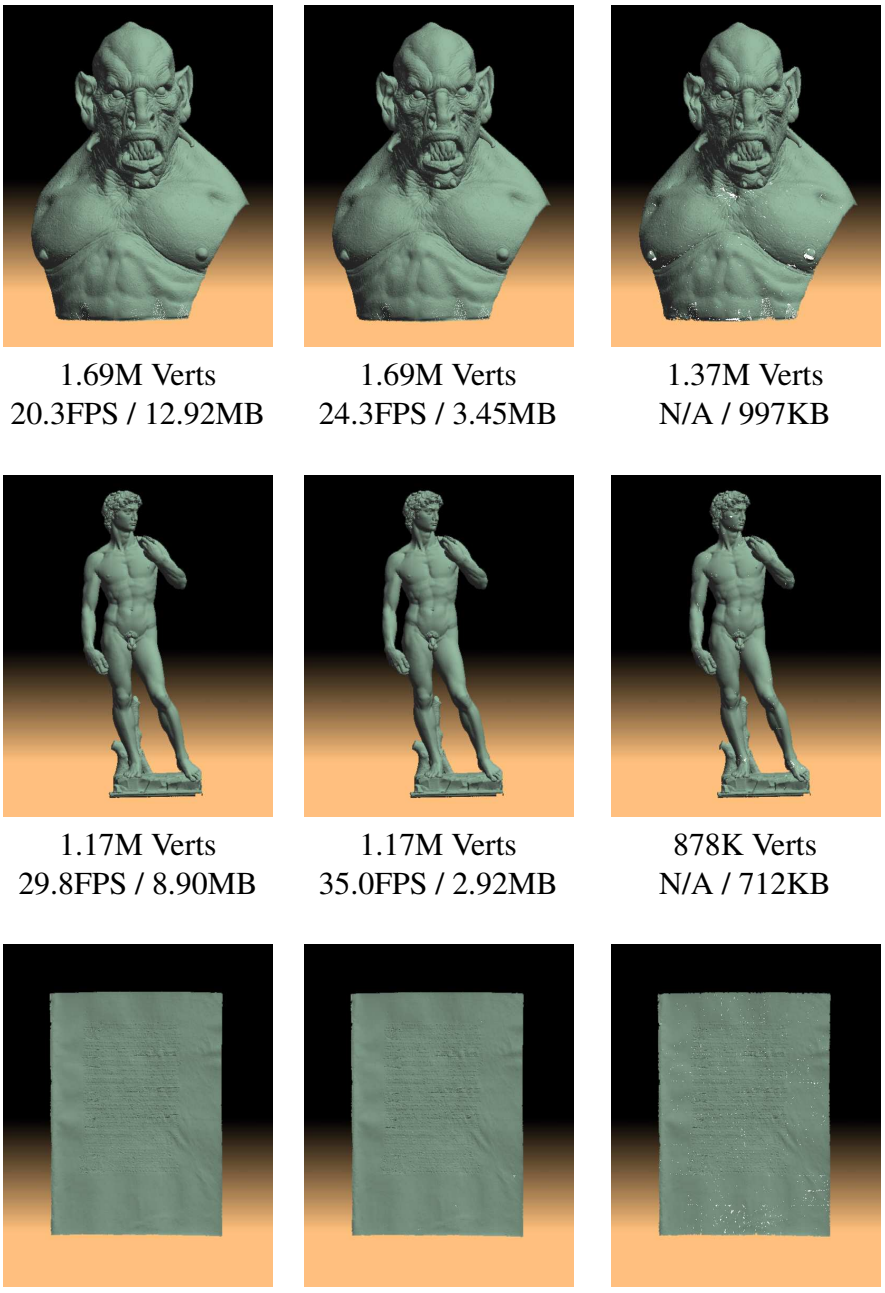


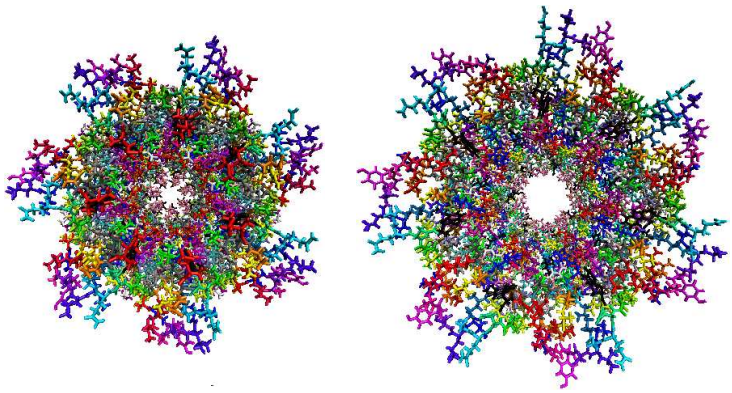
Figure 1.16: *The result of rendering XYZ RGB's Troll, Stanford's David and XYZ RGB's Manuscript. The left images show the conventional rendering of the models, the center images show the rendering of them by salient transformation streams and the right images show the vertices covered by the vertex-transformation pools for the models. We report the number of vertices rendering, the frame rates achieved, and the per-frame communication bandwidth required for the conventional approach and our approach.*

1.7 Salient Frame Detection for Molecular Dynamics Simulations

1.7.1 Problem Definition

Recent advances in acquisition and simulation techniques have generated a huge amount of time-varying datasets. Time-varying data can be acquired from scientific simulation, videos, and animation libraries. Features in the time-varying datasets are commonly defined as the regions of interest that a human observer is likely to look for. As the number and complexity of these datasets increase exponentially [53], it is becoming impractical to expect a human observer or a domain expert to discover all the features manually. Automatic or semi-automatic tools to help humans discover scientifically interesting features are especially important for this reason. To the best of our knowledge, there have been no salient frame detection techniques for molecular dynamics simulations despite a great need for such tools for this area.

Mechanosensitive ion channels play a critical role in transducing physical stresses at the cell membrane into an electrochemical response. The crystal structure of E. *coli* MscS has provided a starting point for detailed descriptions of its mechanism. Figure 1.17 shows the opening of the E. *coli* mechanosensitive ion channel that we will consider throughout Chapter 6. There are 7 subunits in this ion channel, and all 7 subunits are topologically identical, but do not act independently in the simulation. Each subunit has residues 1 to 175 (with few gaps cut out). To understand this mechanism, identifying the presence of kinks in α-helices is critical because they have functional importance.



(a) (b)

Figure 1.17: *Image (a) and (b) show opening of the E.* coli *mechanosensitive ion channel, respectively. There are seven subunits in this ion channel, and all seven subunits are topologically identical, but do not act independently in the simulation.*

1.7.2 Our Approach

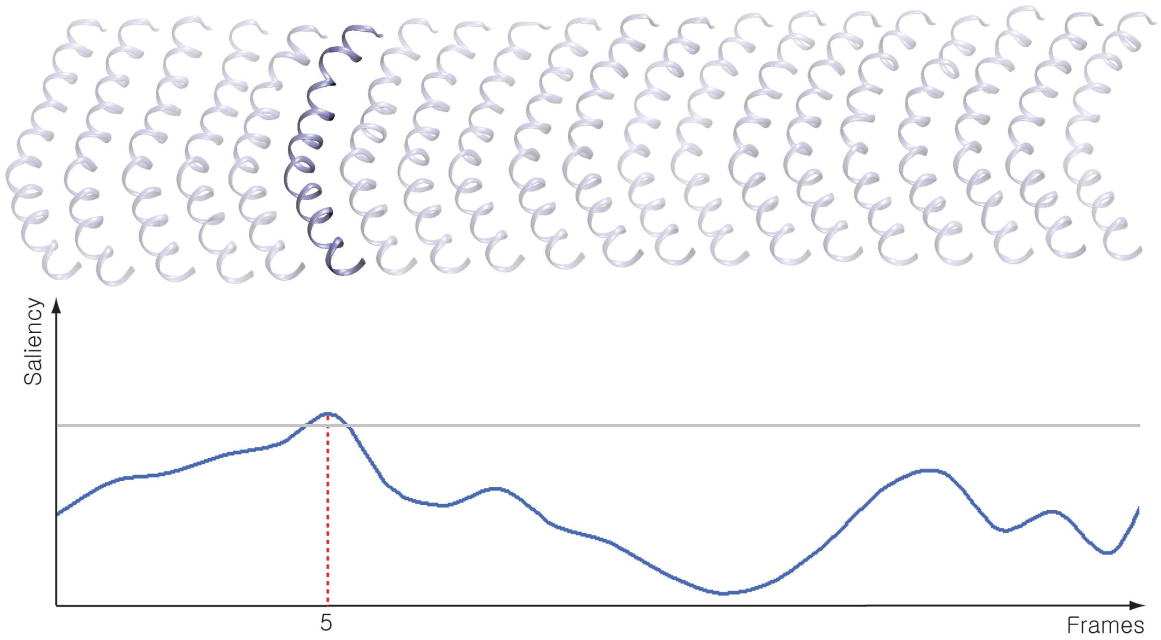
In the method of image saliency by Itti *et al.* [52] or mesh saliency by Lee *et al.* [69], they use a center-surround operator to identify the uniqueness of a pixel or a vertex with respect to its neighborhood. We have decided to use a similar approach and define saliency as the uniqueness of a single frame with respect to its neighboring frames both forwards and backwards in time. Our collaborator (Dr. Sergei Sukharev's group at the Biology Department at the University of Maryland) was interested in identifying the frames in molecular dynamics simulations where the changes in the kinks (anomalies) in the secondary structures occur in the *Ecoli* channel. We validate the effectiveness of our salient frame detection algorithm in this molecular dynamics simulation.

1.7.3 Results of Salient Frame Detection

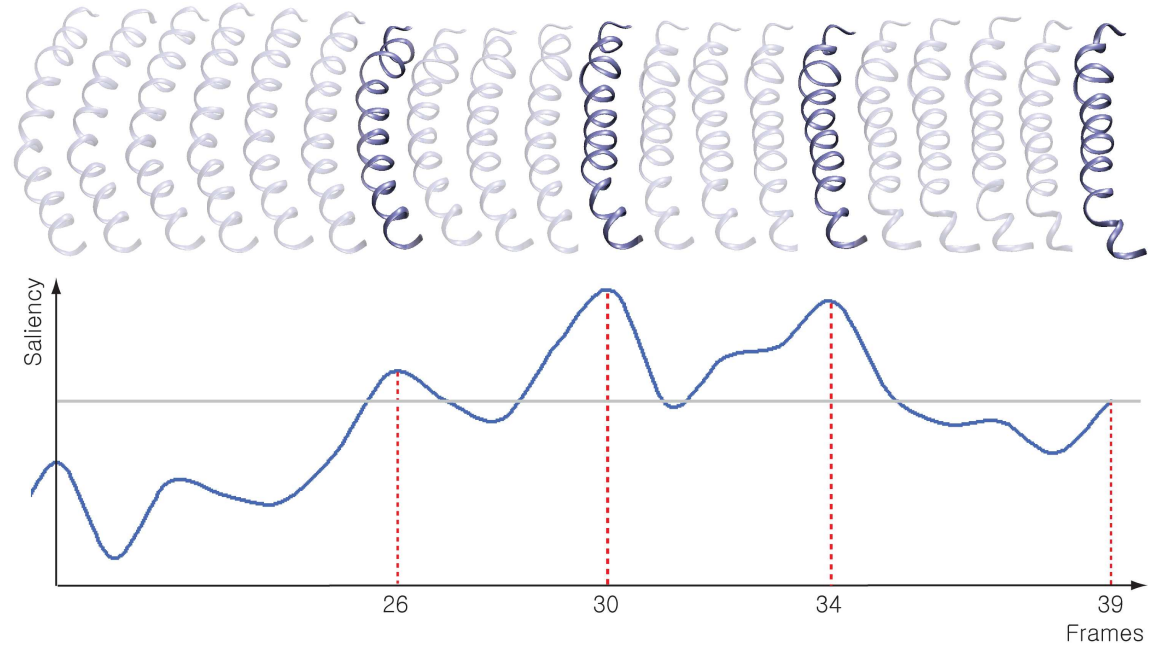
We have compared our detected salient frames with the ones identified independently by our collaborators (biology scientists) for molecular dynamics simulations. Fig-

ure 1.18 shows the five most salient frames detected by our method for the subunit 4 in the E. *coli* mechanosensitive ion channel in Figure 1.17. The frames 5, 26, 30, and 34 which have been detected by our method are the same or very close to the frames 3, 24, 26, 30, and 35 with changes in the kinks, which were detected manually by our collaborators. The frame 39 detected by our algorithm is not close to any frame detected manually by our collaborators, but it had the lowest saliency value among the five most salient frames. Generally, kinks change towards the end of this simulation, and our method successfully detects these important frames.

Figure 1.19 shows the five most salient frames detected by our method for the subunit 1 in the ion channel shown in Figure 1.17. This subunit is topologically identical to the subunit 4, but acts differently in the simulation. Therefore, it results in different salient frames (frames 11, 19, 21, 35, and 39) as shown in Figure 1.19. Our collaborators identified frames 2, 18, 20, 23, 35, 36, and 39 as being salient. Figure 1.20 shows the six most salient frames detected by our method for the subunit 4 in the symmetry annealing of MscS F68S mutant. In this molecular dynamics simulation, residue 68 was mutated to another, serine, which has very specific consequences for channel inactivation in real experiments. As changes in the kinks occur more frequently than the previous simulations, we observe a larger number of salient frames than in the previous cases. Our collaborators have manually identified frames 2, 4, 18, 34, and 38 as being salient. Among these, frames 2, 4, 18, and 38 are the same or close to the frames 1, 5, 18, and 39 detected by our algorithm, and the remaining frame 34 also exhibits a relatively high saliency value as shown in Figure 1.20.

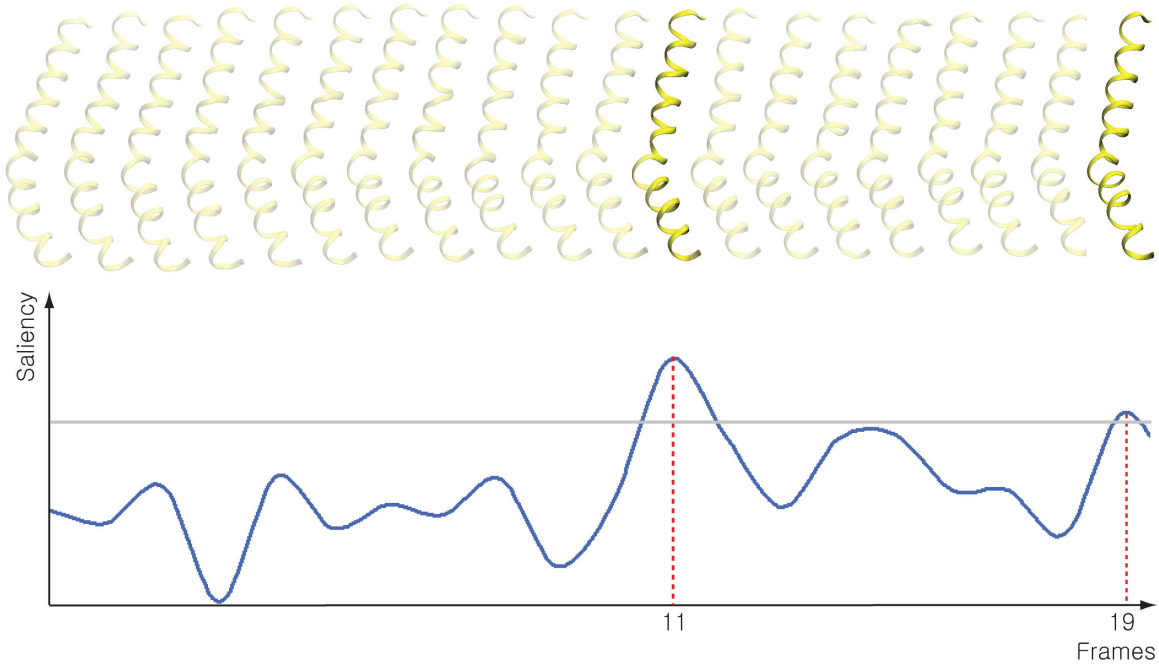


(a) Frames 0 to 19

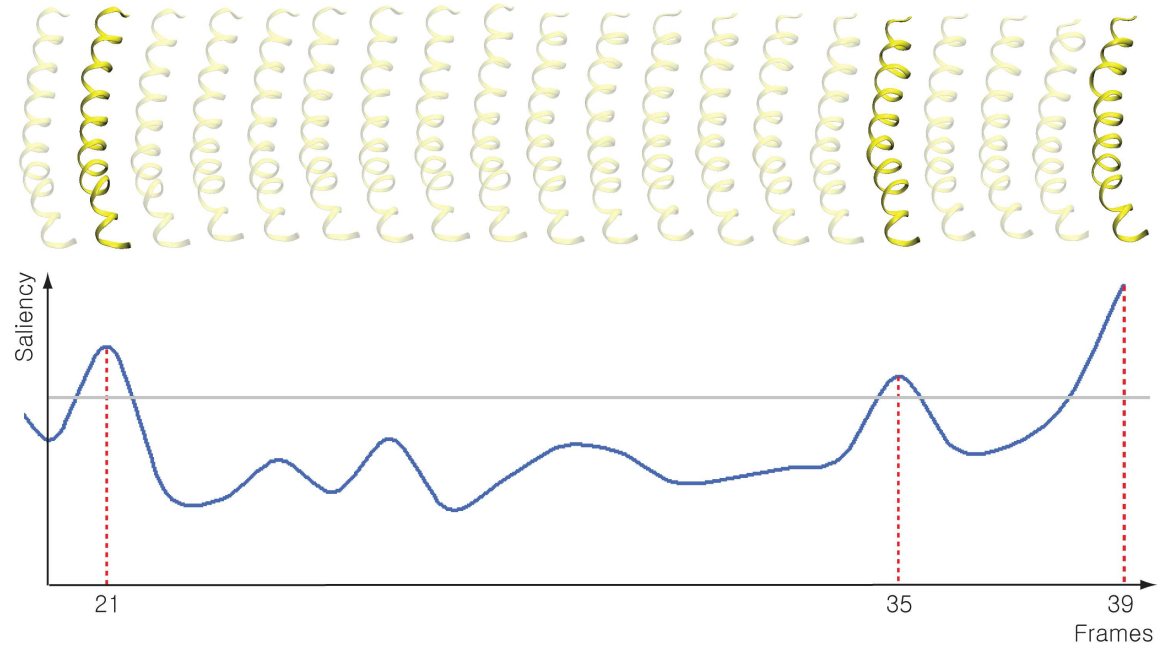


(b) Frames 20 to 39

Figure 1.18: *Five most salient frames detected by our method for the subunit* 4 *in the E.* coli *ion channel (MscS) in Figure 1.17. The changes in the kinks are detected towards the end of this simulation, and our method successfully detects some of the most important frames.*

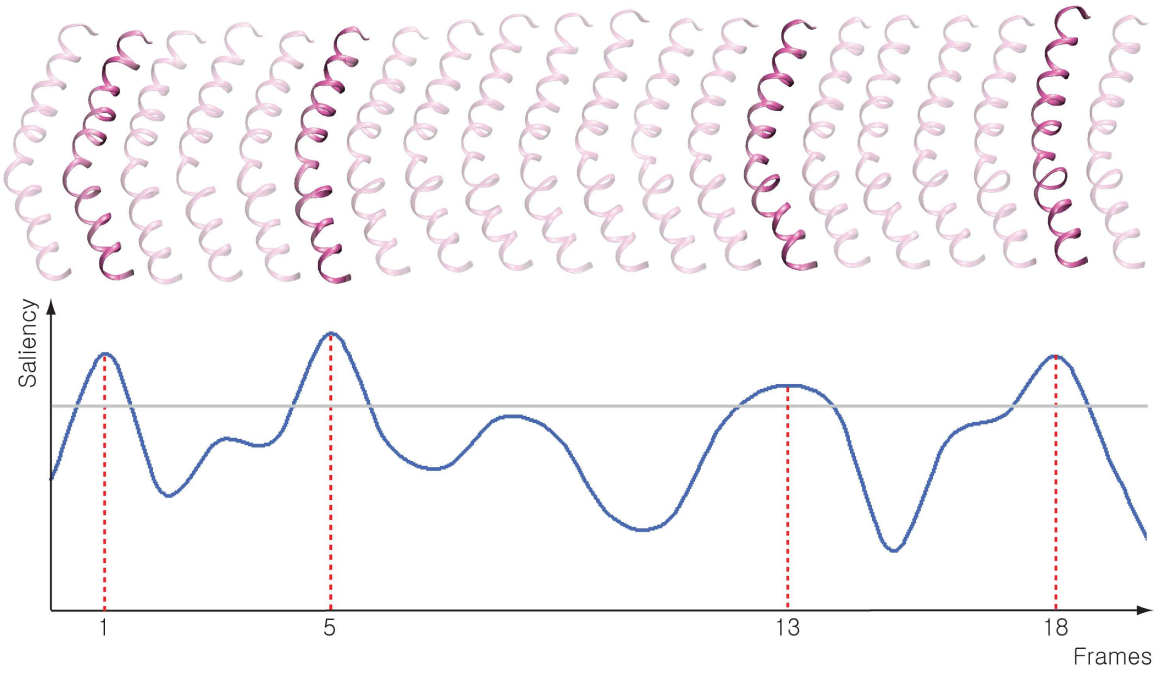


(a) Frames 0 to 19

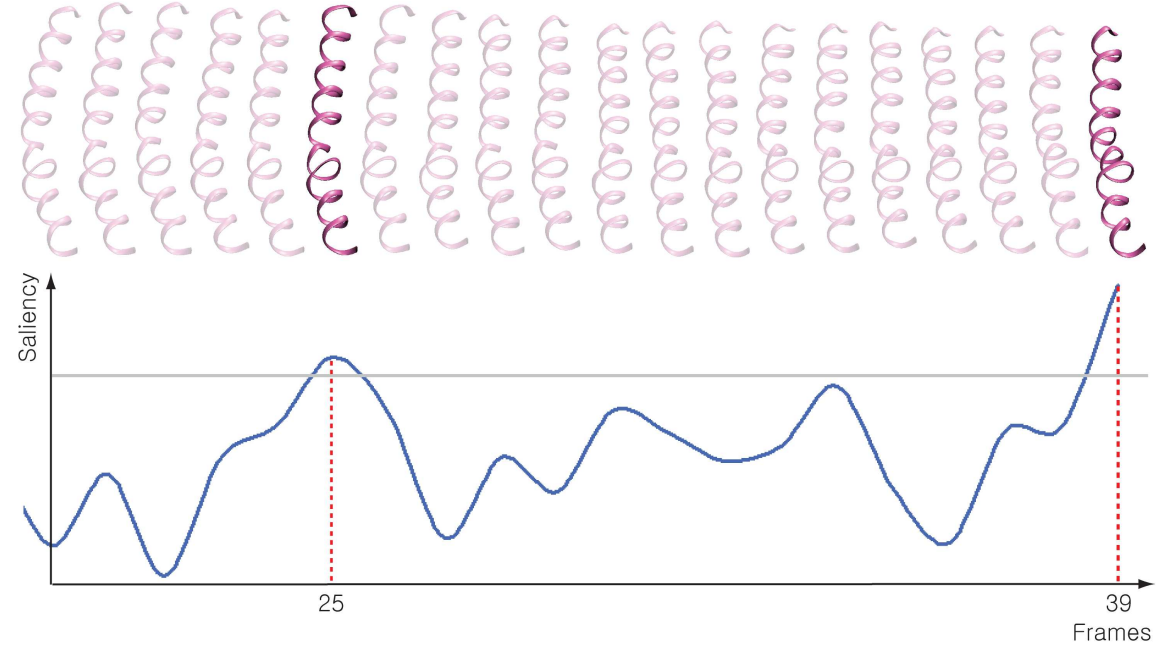


(b) Frames 20 to 39

Figure 1.19: *Five most salient frames detected by our method for the subunit* 1 *in the E.* coli *ion channel (MscS) in Figure 1.17. This subunit is topologically identical to the subunit* 1*, but acts differently in the simulation.*



(a) Frames 0 to 19



(b) Frames 20 to 39

Figure 1.20: *Six most salient frames detected by our method for the subunit* 4 *in the other molecular dynamics simulation, showing the symmetry annealing of MscS F68S mutant – the residue* 68 *was mutated to another, serine, which has very specific consequences for channel inactivation in real experiments.*

Chapter 2

Saliency Validation

Mesh saliency has been proposed as a computational model of perceptual importance for meshes and it has been used in graphics for abstraction, simplification, segmentation, illumination, rendering, and illustration. Even though this technique is inspired by models of low-level human vision, it has not yet been validated with respect to human performance. Here we present a user study that compares the previous mesh saliency approaches with human eye movements. To quantify the correlation between mesh saliency and fixation locations for 3D rendered images, we introduce the normalized chance-adjusted saliency by improving the previous chance-adjusted saliency measure. Our results show that the current computational model of mesh saliency can model human eye movements significantly better than a purely random model or a curvature-based model.

2.1 Related Work

When people examine an image, their eyes tend to fixate on certain points, then jump quickly, with *saccades*, to new points. Although viewers may attend to portions of an image on which they do not fixate, a good deal of evidence suggests that viewers tend to move their eyes to parts of an image that have attracted their attention (see [90], Chapter 11, for a brief review). For this reason, many models of visual attention and saliency have

been evaluated by their ability to predict eye movements. It is not realistic to expect any model to perfectly predict eye movements, because of the variability between human participants and even for the same participant at different times. However, recent research demonstrates that there is a significant correlation between existing models and human eye fixations. For example, Privitera and Stark [99] compare points of fixation by participants to clusters formed by the most salient regions predicted by a large number of simple models of 2D image saliency. They compare this with the degree to which fixations agree between participants. Of the three classes of images they have looked at, they have found that for one class of images (paintings), algorithms based on simple operators including symmetry, center-surround, and discrete wavelet transform cohere very well with human data and approach the coherence among fixations across participants. Parkhurst *et al.* [91] measure the saliency at points of fixation and show that the model of 2D image saliency of Itti *et al.* [52] is more predictive of fixation points than a random model. Previous research also makes the useful methodological points that bottom-up models can better predict the first fixations, which are less influenced by top-down knowledge ([91]), and that the exact order of fixations is highly variable and difficult to predict [99].

There have been many experiments measuring eye movements as participants examine 3D objects [18] [49] [74]. For instance, Cole *et al.* [18] emphasize a region of interest by rendering it in a different style from other regions. They validate the effects of their rendering styles by an eye-tracking experiment. Howlett and O'Sullivan [49] capture saliency values of 3D models by recording where participants look in 3D rendered images. They use this saliency information to simplify 3D models.

The computational model of mesh saliency [69] uses a center-surround mechanism

that is inspired by the human visual system. There have been other approaches to identifying salienct regions on a mesh. Watanabe and Belyaev [127] have identified salient regions on meshes by estimating curvature values, and guided the simplification process to preserve them better. Gal and Cohen-Or [36] have constructed salient geometric features by clustering a set of local descriptors that have a high curvature and a high variance of curvature values for partial shape matching of meshes. Shilane and Funkhouser [111] have identified the important regions of a 3D object which define the object type and distinguish it from other types of objects. They compute how distinctive the regions are with respect to multiple types of objects in a 3D shape database. Feixas *et al.* [31] have defined an information channel between the viewpoints and an object. They use this channel to compute the viewpoint mutual information, which is further used for viewpoint selection and mesh saliency computation. While the notion of saliency in Lee *et al.* [69] and Feixas *et al.* [31] is based on perceptual and visual importance, the notion of saliency in Gal and Cohen-Or [36] and Shilane and Funkhouser [111] is specific to the tasks of shape matching, shape similarity, and shape uniqueness.

In this chapter, we compare a model of mesh saliency with a purely random model and a curvature-based model. There are a number of excellent approaches that generalize differential-geometry-based definition of curvatures to discrete meshes [37, 83, 118, 134]; we use Taubin's method [118] for computing mean curvature in this dissertation.

2.2 Background on User Studies and Psychological Statistics

In this section we present the background on the design of user studies and data analysis using psychological statistics. This will be helpful in understanding our eye-tracking-based user studies performed throughout this dissertation. We refer the interested readers for further details to the seminal work in this area by Cohen [16].

2.2.1 Experiment, Score, Sample, Group, and Variable

Experiments are designed to establish cause-effect relationships. Participants are assigned into two or more groups created by the manipulation of an independent variable (the cause). These groups are measured on the same dependent variable (the effect), returning two or more sets of scores among the groups. Sometimes it is not practical to gather all of the scores from the groups. In such cases, we obtain scores from a subset of the population in the group; this subset is called a sample.

2.2.2 Null Hypothesis and Statistical Significance

The null hypothesis states that the experimental manipulation of an independent variable has no effect on the dependent variable. When we have a certain degree of confidence (say 95%) that the data do not support the null hypothesis, we reject this null hypothesis. Statistical significance means that a given result is unlikely to have occurred by chance. Given a statistical value (such as t-value and F-value) and its corresponding probability (p-value), we conclude whether or not the differences among the scores of the dependent variable in different groups are statistically significant or due to chance.

2.2.3 Degrees of freedom

In the context of fitting statistical models, the term *Degrees of freedom* (df) is the number of independent pieces of information about variability. In other words, it denotes the number of separate pieces of information available to estimate another piece of information. For instance, if you have one score, you have one piece of information about mean, but you do not have any piece of information about variability. If you have two scores, you have two pieces of information about mean and one piece of information about variability because the two scores are equally distant from the mean. Generally, when there are N scores, the degrees of freedom is $N-1$.

2.2.4 t-test

Each sample is characterized by its mean, standard deviation, and a number of scores. We use a t-test to determine if the samples from two groups created by the manipulation of an independent variable could have possibly come from the same distribution (assumed normal). Depending on the degrees of freedom, t-test returns a corresponding p-value. If the p-value is below the threshold chosen for statistical significance usually with the 0.05 or 0.01 level, the means of two normally distributed scores are distinct. There are two versions of the t-test depending on whether the two samples are unpaired or paired. We discuss this next.

Unpaired t-test: In the unpaired t-test, participants are randomly assigned into two groups so that samples are independent of each other. Then the independent groups of participants are compared, and each participant is measured only once on the dependent

variable. The two sets of scores in two groups are therefore uncorrelated.

Paired (Pairwise) t-test: In a paired t-test, each member of one sample has a one-to-one relationship with a particular member of the other sample. When the same participants need to be measured before and after the manipulation of the independent variable, this is the appropriate test to perform. There are still two sets of scores on the dependent variable, but the scores are correlated since the two sets of scores to be compared are obtained from the same participants.

2.2.5 Bonferroni Correction

If we test n hypotheses on a set of data, the probability of being right on all occasions would decrease substantially. For instance, if we test n hypotheses using 0.05 as the significance level, the probability of being wrong at least once increases as $1-0.95^n$. The Bonferroni correction is one of the safeguards against multiple tests of statistical significance on the same data falsely giving the appearance of significance. It says if we test n hypotheses, the statistical significance level should be decreased to $1/n$ times what it would be for one hypothesis. For instance, if there are five independent hypotheses on the same data at 0.05 significance level, instead of using a p-value threshold of 0.05, we would use a stricter threshold of $0.01(=0.05/5)$.

2.2.6 Analysis of variance (ANOVA)

As with the t-test, ANOVA tests for significant differences between groups created by the manipulation of an independent variable. While the t-test is limited to the compar-

ison of only two groups, ANOVA can be used to test differences in three or more groups. The ANOVA produces an F-value and a corresponding probability p-value depending on the degrees of freedom. This probability allows us to reject or retain the null hypothesis. A significant F-value only indicates that there is a significant difference somewhere among the groups; it does not indicate which groups are different. To determine this, secondary comparisons are needed.

One-way ANOVA vs. two-way ANOVA: While one-way ANOVA can only assess the effect of one independent variable on a single dependent variable, two-way ANOVA (or factorial ANOVA) allows us to test the effects of two independent variables. When there are two independent variables, two-way ANOVA is the appropriate test to perform because (1) it generates the same information that two one-way ANOVA's would, and (2) it also assesses the interaction between these variables to influence scores of the dependent variable. While one-way ANOVA only generates one F-value, two-way ANOVA generates three F-values: two to test the main effects of independent variables, and one to test the interaction effect.

Degrees of freedom and Measures of effect size: For ANOVA, two components of degrees of freedom should be calculated: df_{bet} and df_W. The degrees of freedom between groups (df_{bet}) is one less than the number of groups: $\mathrm{bf}_{bet} = k - 1$, and the degrees of freedom within groups (df_W) is equal to the total number of participants minus one for each group: $\mathrm{df}_W = N_T - k$, where k is the number of groups and N_T is the total number of participants.

Measures of effect size in ANOVA are the correlation between the effect (a main effect of an independent variable or an interaction effect) and the dependent variable. It indicates how influential an independent variable or an interaction is on the dependent variable. Partial Eta squared (η^2) is one of the commonly used measures of effect size in ANOVA.

Example: We performed a two-way ANOVA on *saliency values* with two variables: 7 *fixation points* and 2 *saliency models*. *Fixation points* and *saliency models* are independent variables, and *saliency values* is a dependent variable. This analysis gives us three F-values (one to test the main effect of *fixation points*, one to test the main effect of *saliency models*, and one to test the interaction effect of the two independent variables):

- $F(6,238) = 1.066, p = 0.3831, \eta_p^2 = 0.026$ for fixation points
- $F(1,238) = 34.70, p < 0.001, \eta_p^2 = 0.127$ for saliency models
- $F(6,238) = 0.494, p = 0.813, \eta_p^2 = 0.012$ for interaction between two variables

For *fixation points*, the first parameter in the F-value indicates the degrees of freedom between groups (df_{bet}), which is 6 because there are 7 different groups. The second parameter in the F-value indicates the degrees of freedom within groups (df_W). The number of groups is 14 (= 2 (saliency models) $\times$ 7 (fixation points)), and the total number of participants is regarded as 252 (= 14 (groups) $\times$ 18 (participants)). Therefore, df_W (the degrees of freedom within groups) is 238 ($= 252 - 14$). Since the F-value is 1.066 and the corresponding p-value is 0.3831, which is much greater than 0.05, we can say that there is no significant difference among (groups created by the manipulation of) *fixation points*.

For *saliency models*, df_{bet} is 1 because there are 2 saliency models, and df_W is 238 as in the previous case. Since the F-value is 34.70 and the corresponding p-value is less than 0.001, we can say that we can observe significant differences bewteen *saliency models*. The last F-value indicates the ways in which these two variables interact with one another to influence scores of the dependent variable. Since the F-value is 0.494 and the corresponding p-value is greater than 0.05, we can say that there is no interaction between two variables. Partial Eta squared (η_p^2) values also indicate that *saliency models* has the largest main effect on *saliency values* (the dependent variable).

2.2.7 Counterbalancing Problem

When each participant receives more than one manipulation of an independent variable, there exists a possibility of order effects. For example, if the same participant is to see two images (image A and image B) in a row, whichever type of image is presented second will have an unfair advantage (practice) or disadvantage (fatigue) for the participant, depending on the details of the study. The solution is to vary the order in which participants receive manipulations to minimize the bias; this is called counterbalancing. In this example, we can counterbalance the order effects by making half the participants see the image A first and the other half see the image B first. Counterbalancing becomes more complicated when there are more than two manipulation levels. Fortunately, random ordering neutralizes the order effect to some extent, and there are clever counterbalancing schemes, such as the Latin square design, which do not require all possible orders to be presented and yet eliminate order effects.

2.2.8 Differential Carryover Effects

While counterbalancing can eliminate the confounding effects such as practice and fatigue that result from the order of the manipulation levels, it cannot eliminate differential carryover effects. Differential carryover effects are the effects which differ depending on the particular order of manipulations. Suppose we want to measure the effect of the enhancement techniques A and B, and present three similar images (original image, the image A enhanced by technique A, and the image B enhanced by technique B) to participants. If the image A is presented first, it might affect the participants' visual attention for the next image B, thus increasing or decreasing the effect of technique B. This effect may not be symmetrical (e.g., presenting the image B first would not affect a participant's visual attention for the next image A) and would therefore not be balanced out by counterbalancing. Imposing some neutral task or taking time between the presentation of manipulation levels can help reduce differential carryover effects.

2.3 Physical Setup and Fixation Analysis

To gather objective evidence of the correlation between saliency models and human eye fixations, we have carried out two eye-tracking-based user studies (pilot study and main study) and have quantified the similarity between the models and human eye fixations. In this section, we present the physical setup, eye-tracker calibration, and fixation point extraction mechanism which are commonly used in the two studies. We also introduce the *normalized chance-adjusted saliency* to quantify the correlation between a saliency model and fixation points for 3D rendered images. Other experimental design

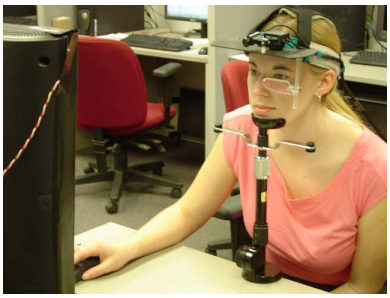


Figure 2.1: *Experimental setup for the user study with the ISCAN ETL-500 eye-tracker.*

issues such as hypotheses, stimuli, and participants of the studies will be explained in Sections 2.4 and 2.5 in detail.

2.3.1 Physical Setup

We used the ISCAN ETL-500 monocular eye-tracker which can record eye movements continuously at 60 Hz. The study was carried out on a 17-inch LCD display with a resolution of 1280×1024, placed at a distance of 24 inches, subtending a visual angle of approximately 31.4 degrees horizontally. The participants had a chin rest to minimize head movements and to maintain calibration. Our experimental setup is shown in Figure 2.1.

2.3.2 Eye-tracker Calibration and Participant Selection

The standard calibration of ETL-500 eye-tracker was performed with 4 corner points and one center point shown in Figure 2.2(a). However, this was not sufficiently accurate for our purposes due to non-linearities in the eye-tracker-calibrated screen space. Therefore we used the second calibration step which involves a more densely-sampled calibration phase similar to [91] with 13 additional points shown in Figure 2.2(b). For

this we asked the participants to successively look at and click on 13 points presented on the screen. This gave us an accurate correspondence between the eye-tracker space and the monitor space for that participant. We then triangulated the monitor's screen space using these 13 points and 4 corner points from the first phase calibration as shown in Figure 2.2(c). Such a triangulation allowed us to get an accurate position on the monitor by interpolating inside the triangle where the subject was looking. After this we tested the accuracy of the calibration by asking the participants to look at 16 randomly selected

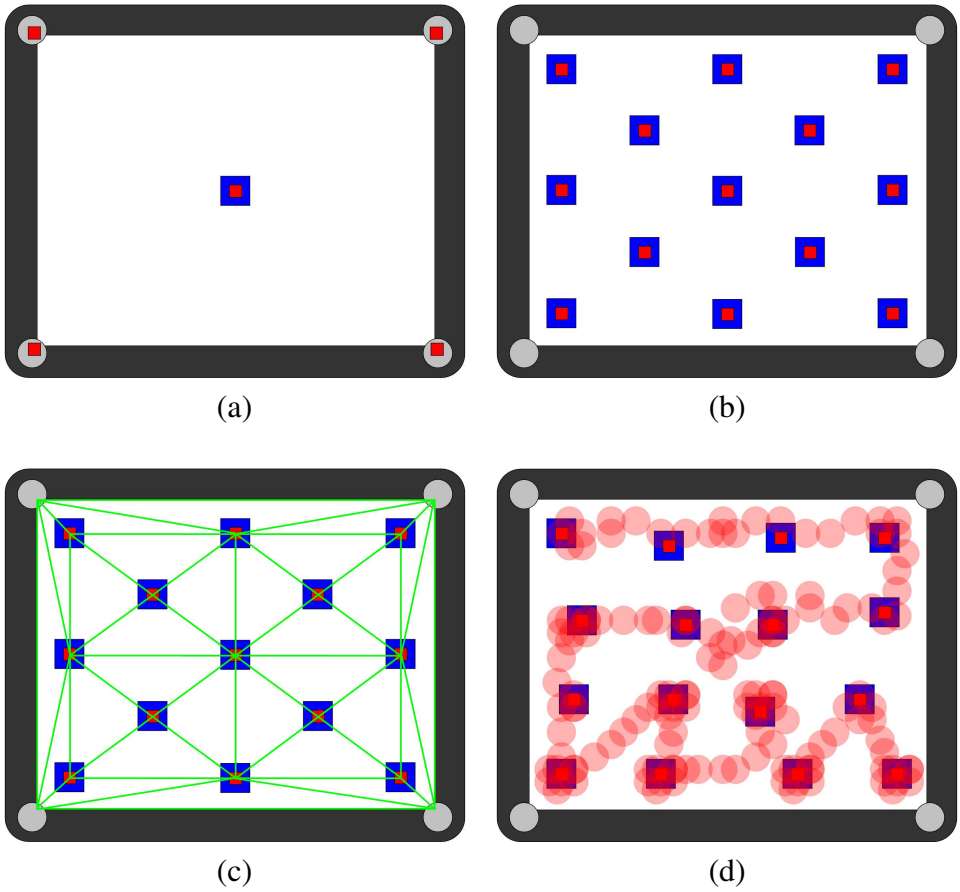
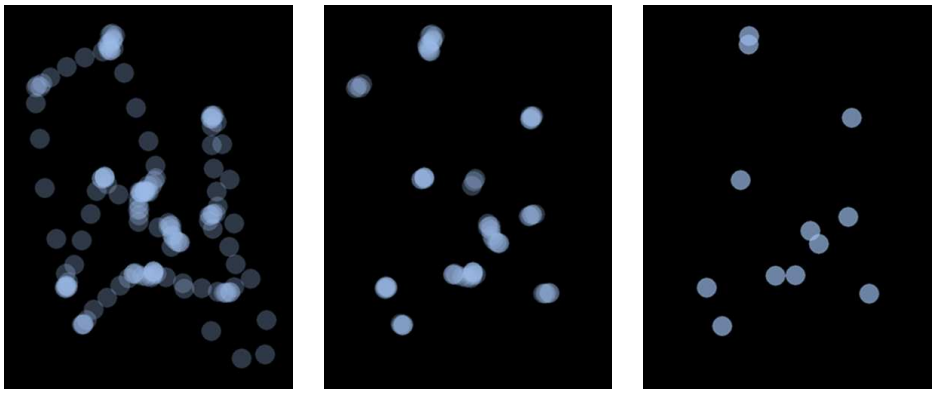


Figure 2.2: *Eye-tracker calibration steps used in our study. Image (a) shows 4 corner points and one center point used for the standard calibration of ETL-500 eye-tracker. Image (b) and (c) shows 13 additional points used in our second calibration step and triangulation based on these, respectively. Image (d) shows eye movements on 16 randomly selected points to test the accuracy of the calibration.*

points (shown in Figure 2.2(d)) on the screen. We proceeded with our study using the participants who were able to successfully calibrate to within an accuracy of 30 pixels (about .75 degrees) for each of the 16 points. Our participants had normal or corrected-to-normal vision and were not familiar with the goals of this study. The participants were told to freely view the images with no assigned goal.

2.3.3 Fixation Points

We divide the raw data points from the eye-tracker into two groups – *fixation points* which correspond to a user looking at a single location and *saccade points* which correspond to fast eye movements from one fixation point to the next. We followed an approach similar to the one suggested by Stampe [113] to identify fixations and saccades. Figure 2.3 shows a two step process to extract fixation points from the raw data points. We considered data points that had a velocity greater than $15^{\circ}/sec$ as saccade points and removed them. We then averaged consecutive eye locations that were within 15 pixels and classified them as a single fixation point. Some researchers have advocated discarding short (exploratory) fixations in measuring the attention of the viewer [47] to discriminate between distraction and attention. We ignored brief fixations below the threshold of 133ms. This corresponds to 8 consecutive points in the ISCAN ETL-500 eye-tracking device.



(a) Raw Data Points (b) Saccades removed (c) Fixation Points

Figure 2.3: *Image (a) shows all the raw data points from the eye-tracking device. Image (b) shows the points remaining after removing saccade points. Image (c) shows final fixation points after removing brief fixations and combining consecutive points if they are spatially close.*

2.3.4 Normalized Chance-adjusted Saliency

2.3.4.1 Chance-adjusted Saliency

Parkhurst *et al.* [91] introduced the notion of chance-adjusted saliency to quantify the correlation between stimulus saliency and fixation locations for an image. They compute the chance-adjusted saliency as follows. Consider a collection of images I_i, $1 \leq i \leq N$. A participant is asked to look at each image in turn. This generates a set of fixation points f_{ij}, $1 \leq j \leq F_i$ for each image I_i, where F_i is the number of fixation points. Let us consider the k-th fixation points f_{ik} across all the images I_i. Let s_{ik} be the saliency value at the k-th fixation point f_{ik} in the image I_i. They compute the mean fixation saliency for the k-th fixation points as $\bar{s}_k^f = \frac{1}{N}\sum_{i=1}^{N} s_{ik}$. To compute the mean random saliency, they first generate F_i random points r_{ij} over each image I_i, where $1 \leq i \leq N$ and $1 \leq j \leq F_i$. Then, the mean random saliency $\bar{s}_k^r$ is computed as the average saliency over the k-th random point r_{ik} across all the images I_i, $1 \leq i \leq N$. Finally, they define

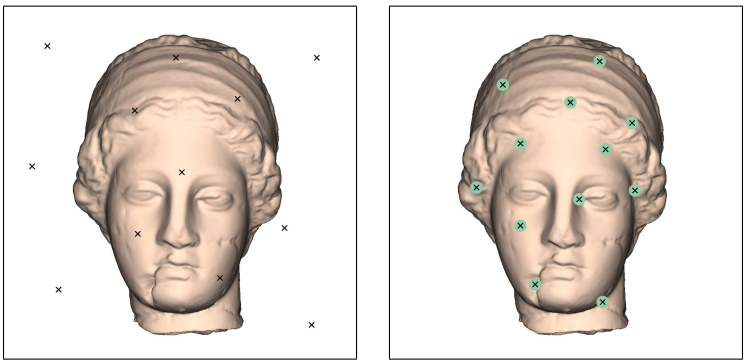


Figure 2.4: *The left image shows the random points in chance-adjusted saliency computation. These points are scattered all over the image. The right image shows the points that we consider in normalized chance-adjusted saliency computation. We only include the foreground pixels that are covered by projected triangles of the mesh. For each fixation point represented as a cross, we also take into account the eye-tracker accuracy of* 20 *pixels, which is represented as a circle.*

the chance-adjusted saliency (s_k^c) for the k-th fixation points as the difference between the mean fixation saliency ($\bar{s}_k^f$) and the mean random saliency ($\bar{s}_k^r$): $s_k^c = \bar{s}_k^f - \bar{s}_k^r$.

2.3.4.2 Normalized Chance-adjusted Saliency

We observed three shortcomings in using the previously defined chance-adjusted saliency to quantify the correlation between human eye fixations and the model of mesh saliency.

1. The chance-adjusted saliency was developed for images in which there is a well-defined saliency at every pixel. We are trying to measure the correlation between a mesh saliency approach and the fixation points on the mesh but not the fixations on the entire rendered image. Therefore, we should only consider the foreground pixels that are covered by projected triangles of the mesh. This ensures fairer comparisons between a random model and the saliency model for 3D rendered images because excluding the background pixels would prevent lowering the

average saliency values in a random model. Figure 2.4 shows the points considered in chance-adjusted saliency and normalized chance-adjusted saliency.

2. The chance-adjusted saliency does not consider eye-tracker accuracy. Since the fixation point acquired from the eye-tracker can differ from the actual pixel that a user looked at, we have to consider the eye-tracker accuracy as shown in Figure 2.4(b) when we assign the mesh saliency value to the fixation point.
3. The chance-adjusted saliency is defined over a collection of images. This restricts the analysis of the effect of different models. We need a method that normalizes saliency on a per-image basis.

To address these problems, we define *normalized chance-adjusted saliency* in this section. First, we consider the eye-tracker accuracy ε which depends on both the accuracy of the eye-tracking device and the calibration steps. We have used $\varepsilon = 20$ pixels, subtending a visual angle of approximately 0.5 degree horizontally. Note that a fixation point and a pixel share the same coordinate system. Let us consider the pixel p_{ij} on which a fixation point f_{ij} falls. Instead of taking the saliency value on a fixation point f_{ij}, we compute the error-adjusted saliency s_{ij}^{ε} as the maximum of the saliency values within a radius of $\varepsilon = 20$ pixels around p_{ij} in the image I_i, $1 \leq i \leq N$: $s_{ij}^{\varepsilon} = \max_{k \in \mathcal{N}_j^{\varepsilon}} s_{ik}$, where $\mathcal{N}_j^{\varepsilon} = \{k | dist(p_{ij}, k) \leq \varepsilon\}$. For each rendered image I_i, we compute the mean ($\bar{s}_i^{\varepsilon}$) of the saliency s_{ij}^{ε} over all the pixels j that are covered by the rendered mesh. We now define our normalized chance-adjusted saliency for the fixation point f_{ik} as $s_{ik}^{n} = s_{ik}^{\varepsilon} / \bar{s}_i^{\varepsilon}$. Here we use the ratio instead of the difference to enable it to be used across different models and different view points; otherwise we will need to normalize the 3D saliency values for each rendered image for fair comparisons with different models and different views of

a model. We note that we could have computed the mean of the error-adjusted saliency differently for each participant as in chance-adjusted saliency: generate random points on the pixels covered by the rendered mesh and compute the average saliency values on these random points. However, this causes high variance in the means among different participants, resulting in a high variance in normalized chance-adjusted saliency values even for the same image. To avoid this high variance, we have decided to use $\bar{s}_i^{\epsilon}$ in computing normalized chance-adjusted saliency for all fixation points on the image I_i.

2.4 Pilot Study

To figure out what independent variables can affect the correlation between saliency models and human eye fixations, we performed a pilot study with 6 subjects. In this section, we would like to summarize its experimental design and some of the interesting results. We will also explain what we have learned from our pilot study and how these lessons affected our design choices for the main study explained in Section 2.5.

2.4.1 Stimuli

We have illustrated five natural scanned models used for our study in Figure 2.5. Each model was shown from 10 different views. Since the computational model of mesh saliency relies only on geometric properties (curvature values), we would like to see whether it actually correlates with the human eye fixations from all viewing directions. For this purpose, we generated images from 10 different views and used them in our study. As shown in Figure 2.6, we have generated ten (five right-side-up and five upside-down)

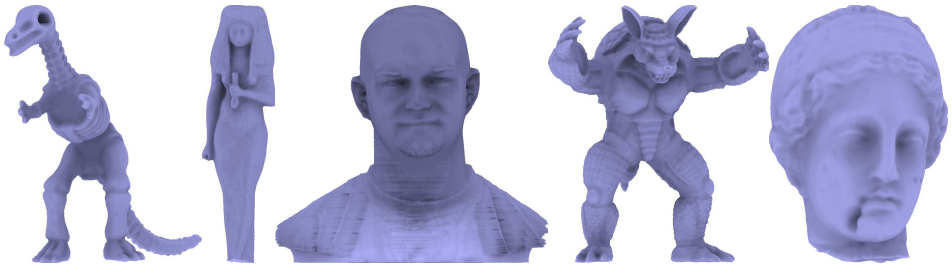


Figure 2.5: The Dinosaur, Isis, Male, Armadillo, and Igea models used in our pilot study.

views for each model. We manually choose the first view. We rotate this model $-30°$, $-15°$, $15°$, and $30°$ along the vertical axis to generate four more views of the model. Then we turn each of these views upside down to generate the remaining five views.

Image Ordering: The user study had 52 trials (images). The data captured for the first two images was discarded as they were intended to give the subjects a sense of the duration. Each trial started with the subject seeing a blank screen with a cross at the center of the screen. The subject was asked to look at the cross before clicking the mouse to bring up the next image. This ensured that each trial started with the subject's eyes fixated at the center of the image. Each image was shown for 5 seconds. When we ordered the images for each user, we minimized differential carryover effects by placing similar images far apart. Alleviating differential carryover effect was very important because each user looked at 5 similar images ($-30°$, $-15°$, $0°$, $15°$, and $30°$).

Image Synthesis Consideration: In our pilot study for validating mesh saliency, we wanted to minimize the influence of lighting on the human perception of the rendered images. The easiest solution is to use a simple ambient term, but this approach leads to indiscriminate flatness. Instead, we use ambient occlusion [67] [137], in which illumination at a vertex is proportional to the fraction of the environment that it can see. We

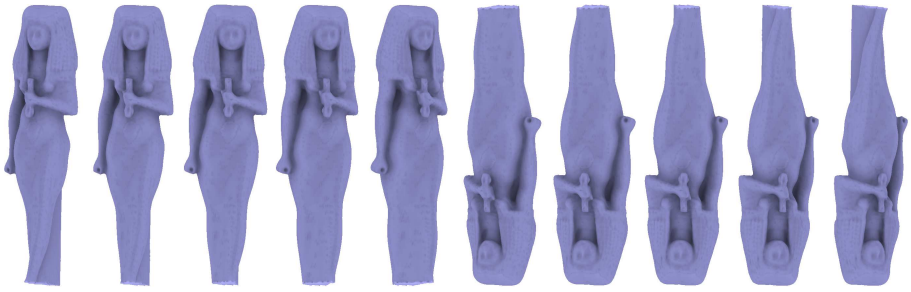


Figure 2.6: Ten different views of the Isis model with ambient occlusion. There are five right-side-up and five upside-down views, and these views are rotated 15 degrees apart along the vertical axis.

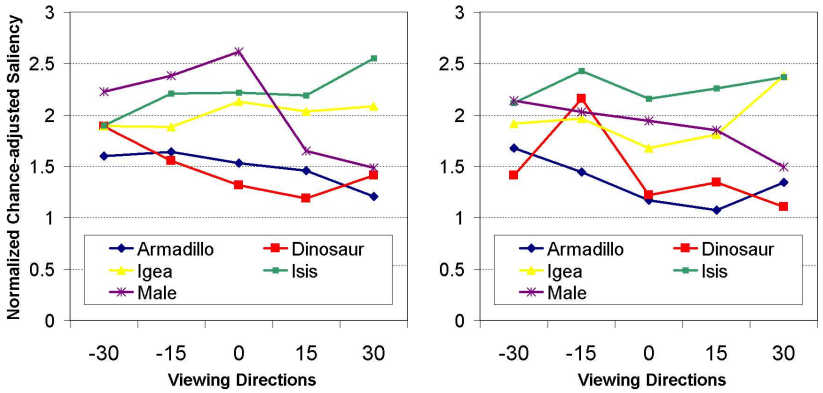
preprocess the view-independent occlusion information using ray casting and then use this information at runtime for ambient occlusion lighting.

2.4.2 Hypothesis

The hypothesis in our pilot study is that the computational model of mesh saliency has better correlation with human eye fixations than a random model regardless of viewing direction for the first few seconds after stimulus onset. There are three independent variables in our pilot study: models, rotations of models, and right-side-up vs. upside-down views.

Models and rotations: Parkhurst *et al.* [91] have observed that fixations for the subjects are usually biased towards the center. In our experiment, each subject is asked to look at the cross at the center of the screen before each trial. By using different models and rotating the models, we change the distances from the center to the high saliency regions.

Right-side-up vs. upside-down views: Recent work on gaze control has focused on two attentional models: bottom-up stimulus-based information and top-down



(a) Right-side-up Views (b) Upside-down Views

Figure 2.7: Average normalized chance-adjusted saliency value across subjects for each viewing direction for each model. For all the cases, the values are higher than 1, which is the value that can be expected by chance.

memory-based knowledge [46]. The 3D models we used in our study were not absolutely knowledge-free as they were scanned models of animals or humans. Subjects could use generic semantic and spatial knowledge even though we are measuring their eye movements for short time (the first five seconds). Parkhurst *et al.* [91] support this argument by showing that stimulus dependence is greatest for early fixations in their work. We include upside down views to slow down the onset of top-down attentional effects.

2.4.3 Results

The results of our normalized chance-adjusted saliency values can be seen in Figure 2.7. In general, we have observed that the computational model of saliency has a higher correlation with human eye fixations than a random model as the normalized chance-adjusted saliency values are higher than 1, the value that can be expected purely by chance. However, there is considerable variability depending on the models and rotations of models as shown in Figure 2.7.

To observe the effects of each independent variable, we first carried out a three-way ANOVA on the normalized chance-adjusted saliency values with three conditions: different models (M), rotation angles (A), and right-side-up vs. upside-down views (I). For models, we have observed significant differences: $F(4,250) = 55.424, p < 0.001, \eta_p^2 = 0.917$ as expected in Figure 2.7. For rotation angles, we have also observed significant differences: $F(4,250) = 5.012, p < 0.006, \eta_p^2 = 0.501$. The result indicates that even though saliency of each vertex in a model can be computed once and does not change, the projection of a model could result in different behavior in drawing viewers' attention depending on viewing directions. The possible causes include occlusion amongst salient and non-salient regions and distance changes from the center to the salient regions. For right-side-up vs. upside-down views, we have observed a marginal difference ($F(1,250) = 4.8664, p = 0.079, \eta_p^2 = 0.493$). Though the difference is not significant ($p > 0.05$), the effect size index (η_p^2) suggests that there is a moderate difference here and this non-significant result is probably caused by a small number of subjects. Turning the model upside down is likely to reduce the effect of semantics, and therefore this result indicates the effect of semantics plays a certain role in the correlation between eye fixations and the mesh saliency model in our study. Since there were three independent variables (M, A, and I), we investigated the interaction effects between any two independent variables (M and A, M and I, and A and I). Among these, only the interaction effect between model and rotation angle was significant: $F(16,250) = 4.428, p < 0.001, \eta_p^2 = 0.470$. This means that the effect of rotation angles depends on the model that we consider.

We have also carried out a pairwise t-test on the average saliency values between fixation points and random points for each model (this is the only condition in the test).

Table 2.1: *List of pairwise t-tests (two-tailed).*

Model	t-Value	p-Value
Armadillo	-8.37	< 0.0001
Dinosaur	-7.20	< 0.0001
Igea	-18.26	< 0.0001
Isis	-17.18	< 0.0001
Male	-12.54	< 0.0001

We compared the model of saliency to the random model as in Parkhurst *et al.* [91]. We compute the average saliency values from randomly chosen locations lying on the foreground instead of the observed fixation locations. Table 2.1 shows that there are significant differences in the average saliency values for all the 3D models between the mesh saliency model and the random model. The results validate that the mesh saliency model has significantly higher correlation with human eye fixations than a random model regardless of viewing direction.

2.4.4 Limitations

The results in our pilot study show that the mesh saliency model has significantly higher correlation with human eye fixations than a random model regardless of the viewing direction. In general, the number of subjects in our pilot study was too small to draw meaningful conclusions in several places. For example, in analysis of the effect of right-side-up vs. upside-down views, we have observed a marginal difference ($F(1,250) = 4.8664, p = 0.079, \eta_p^2 = 0.493$). When we performed another two-way ANOVA with rotation and right-side-up vs. upside-down views for each model, we

could also observe the marginal differences in normalized chance-adjusted saliency values for the Armadillo and the Male models ($F(1,50) = 5.562, p = 0.065, \eta_p^2 = 0.527$ and $F(1,50) = 6.581, p = 0.050, \eta_p^2 = 0.568$, respectively). The interactions between rotation angles and right-side-up vs. upside-down views were the strongest for these two models as well. This might support the idea that there were some other effects, such as those due to semantics, when subjects looked at these models. However, this is still questionable, since the difference between right-side-up views and upside-down views was not statistically significant ($p \geq 0.05$).

All of these limitations and other concerns in our pilot study made us modify our experimental design in the following ways:

1. For better analysis of results, we performed our new experiment with 18 subjects and analyzed the results.
2. We realized that the use of ambient occlusion made the images unnatural even though they could minimize the effect of local lighting. We instead used a local lighting model with a directional light source from a viewer's position in our new study.
3. While the effects of viewing direction suggested by our pilot study are intriguing, a thorough analysis of this issue requires careful study, which we leave for future work. In addition to the main effect of rotation angles caused by occlusion and distance changes explained in Section 2.4.3, the local lighting model could completely hide some salient features on the surface of a model depending on the normal and the lighting directions. We need to design a new study much more carefully by considering all the implications of view dependency.

4. We included a curvature-based model and compared the mesh saliency model to this as well as a random model.

2.5 Main Study

Our pilot study has shown that there may be certain view-dependent effects such as view angles and orientation on human eye movements. Although studying view-dependence of mesh saliency is an important area, a view-independent model of mesh saliency is desirable for several applications. For instance, offline mesh simplification, mesh segmentation, view selection, lighting design, as well as allocation of computational resources in the rendering pipeline all benefit from a view-independent model of mesh saliency. For such applications, it is crucial to establish the correlation between human eye movements and view-independent models of mesh saliency.

Considering all the limitations discussed in Section 2.4.4, we performed another user study with 18 subjects in the slightly different setting. We will explain our new experimental design by focusing on the changes from the pilot study in this section.

2.5.1 Stimuli

We have used the same set of 3D models illustrated in Figure 2.8. Each image was shown for 5 seconds and we randomized the order of images to counterbalance overall effects. There are two differences from the pilot study: (i) We only consider the front view of each model in our main study. (ii) We use a local lighting model with a directional light source from a viewer's position.

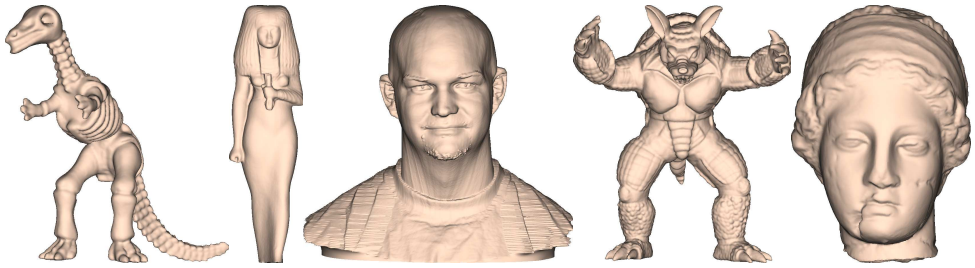


Figure 2.8: *The Dinosaur, Isis, Male, Armadillo, and Igea models used in our study.*

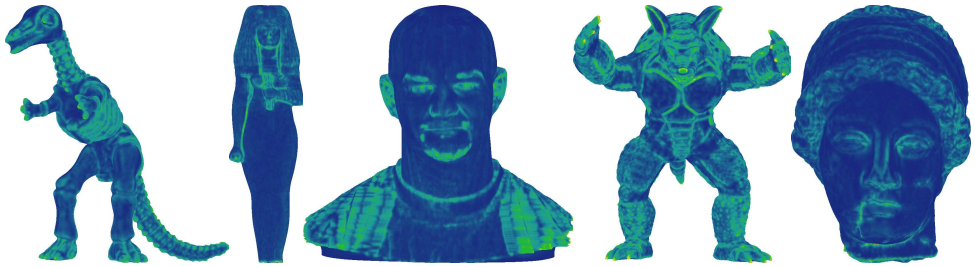
2.5.2 Hypothesis

Our hypothesis in our main study is that the computational model of mesh saliency has better correlation with human eye fixations than a random model and a curvature-based model for the first few seconds after stimulus onset.

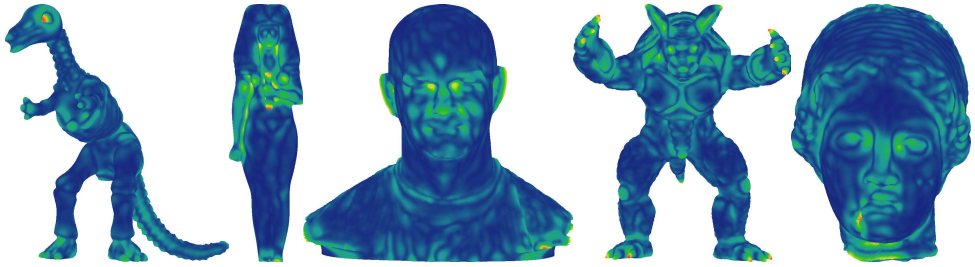
2.5.3 Results

Figure 2.9 shows the fixation points and computed mean curvature and mesh saliency for each model. Fixation points are illustrated with hot spot maps, where warm colors show highly fixated regions. We observe that most fixations are close to warm-colored salient regions computed by the model of mesh saliency.

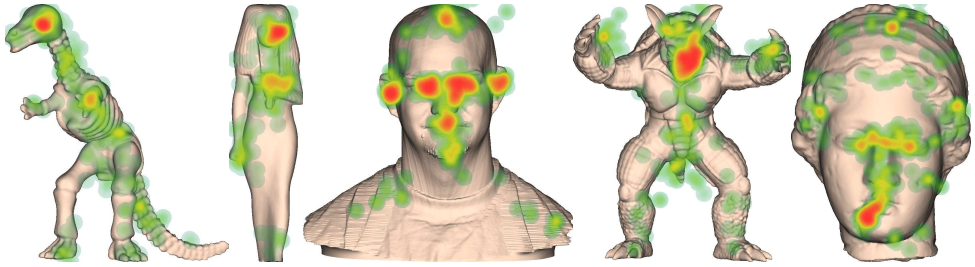
We first report the results of chance-adjusted saliency values and identify some shortcomings along the lines of what we discussed in Section 2.3.4. We then present the results using our new normalized chance-adjusted saliency values. Figure 2.10 shows the average of the chance-adjusted saliency values across all participants. As in [80, 88], the study is restricted to the first seven fixations. In general, we observe that the curvature-based model and mesh saliency model have a higher correlation with human eye fixations than a random model as the chance-adjusted saliency values are higher



(a) Computed Mean Curvature



(b) Computed Mesh Saliency



(c) Human Eye Fixations

Figure 2.9: *The saliency models and human eye fixations. The first and second rows show the mean curvature and mesh saliency, respectively on the models used in the study. Here warm colors indicate high saliency regions while cool colors indicate low saliency regions. The third row shows the human eye fixations from our eye-tracking-based user study with hot spot maps.*

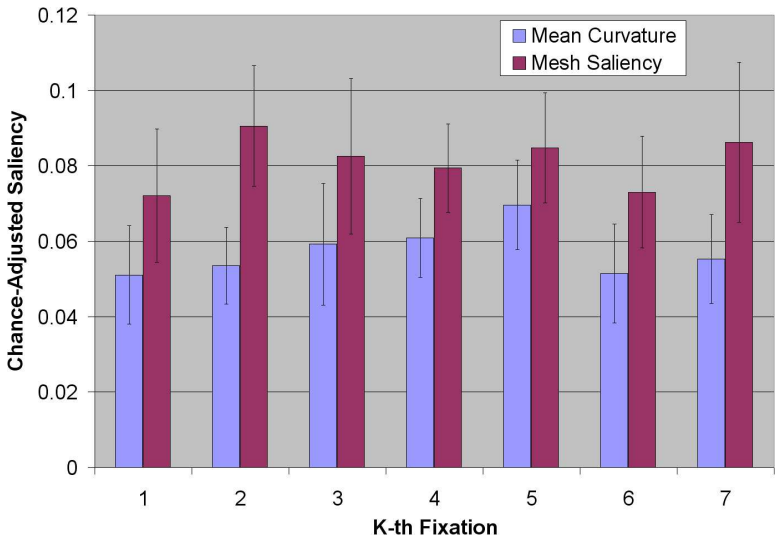


Figure 2.10: *Average chance-adjusted saliency values and 95% confidence interval using curvature and mesh saliency across all participants for each model. For all the cases, the values are higher than 0, which is the value that can be expected by chance. They exhibit high variances in computed chance-adjusted saliency values for all cases.*

than 0, the value that can be expected purely by chance. As noted in Section 2.3.4, one of the problems in chance-adjusted saliency is that it can be overvalued if random points on background pixels are included in the process of computing the mean random saliency. This overvaluation causes high variances in computed chance-adjusted saliency for all k-th fixation points ($1 \leq k \leq 7$) in Figure 2.10. We have performed a two-way ANOVA on the chance-adjusted saliency values with two variables: k-th fixation points ($1 \leq k \leq 7$) and different saliency models. For k-th fixation points, there is no significant difference ($F(6,238) = 1.066, p = 0.3831, \eta_p^2 = 0.026$). However, for saliency models (curvature model and mesh saliency model), we observed significant differences ($F(1,238) = 34.70, p < 0.001, \eta_p^2 = 0.127$). There was no interaction between two variables ($F(6,238) = 0.494, p = 0.813, \eta_p^2 = 0.012$).

Figure 2.11 shows the average normalized chance-adjusted saliency values computed by the curvature model and the mesh saliency model across all participants. In

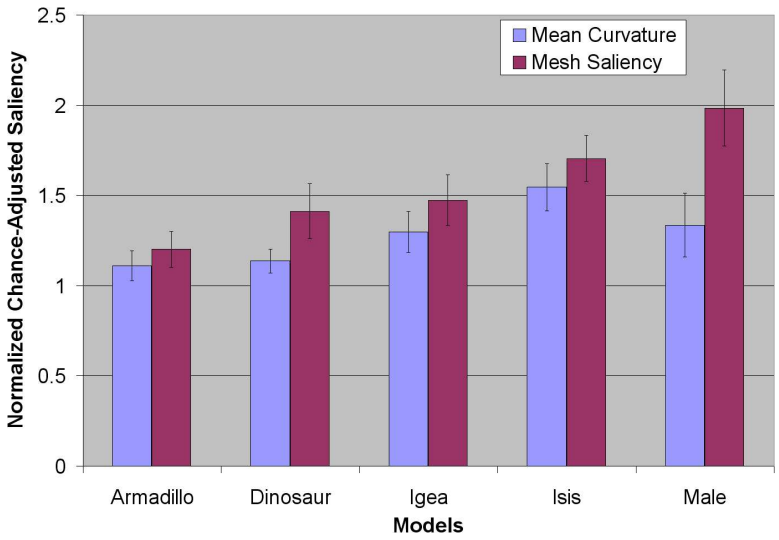


Figure 2.11: *Average normalized chance-adjusted saliency values and 95% confidence interval using curvature and mesh saliency across all participants for each model. For all the cases, the values are higher than 1, which is the value that can be expected by chance.*

general, we have observed that both computational models of saliency have higher correlation with human eye fixations than a random model as the normalized chance-adjusted saliency values are higher than 1, the value that can be expected purely by chance. We observe lower variances in normalized chance-adjusted saliency than chance-adjusted saliency cases. To observe the effects of saliency models and 3D models, we carried out a two-way ANOVA on the normalized chance-adjusted saliency values with two variables: different saliency models and different 3D models. We have found that there is a main effect of saliency models ($F(1,170) = 37.75, p < 0.001, \eta_p^2 = 0.182$). This indicates that the mesh saliency model exhibits higher correlation with human eye fixations than the curvature-based model. A significant main effect was also obtained for 3D models ($F(4,170) = 19.86, p < 0.001, \eta_p^2 = 0.319$). The result indicates that there are certain models which correlate better with human eye fixations than others. The possible cause is the difference in numbers of highly salient regions inherent in models. For instance, the

Table 2.2: *The Pairwise t-tests (two-tailed)*

Model	t-Value	p-Value
Armadillo	-2.643	0.017
Dinosaur	-4.122	< 0.001
Igea	-4.546	< 0.001
Isis	-3.786	0.001
Male	-7.489	< 0.001

Armadillo model has a large number of highly salient regions distributed on the model while the Male model has a small number of highly salient regions as one can observe in Figure 2.9. We have also found there is a strong interaction between two variables ($F(4,170) = 5.093, p = 0.01, \eta_p^2 = 0.107$), meaning that the effects of saliency models depend on 3D models under consideration. Next, we perform a pairwise t-test on the normalized chance-adjusted saliency values between two saliency models (curvature-based and mesh saliency). Table 2.2 shows the result for each of the models. We found a significant difference between two saliency models in the normalized chance-adjusted saliency values for each of the 3D models. Even with the Bonferroni correction, we found a borderline statistical significance for the Armadillo model and statistically significant results for all other models.

The results validate that the mesh saliency model has significantly higher correlation with human eye fixations than a random model and a curvature-based model.

2.6 Discussion and Future Work

We have used a few devices to reduce the effect of semantics: (i) we did not give any tasks to the users when viewing images, and (ii) we limited the time of a stimulus to the first five seconds. Others [91] [108] have also used similar durations. However, five seconds could be considered too long since semantic interpretation starts increasing right after the stimulus onset. Further study is needed to quantify the effect of semantics with varying duration. Another thing we can do to reduce the effect of semantics is to experiment with objects that do not carry semantic information for most users (such as molecular models) or close-up views of scanned models.

We currently compute mesh saliency in a view-independent way. However, there is some evidence in our pilot study showing that the correlation between eye fixations and mesh saliency is view dependent. Further study is needed to fully understand the implication of view-dependent variables such as illumination and viewing angles on visual saliency. In this context, it will be interesting to compare and contrast the mesh saliency model to 2D image saliency models.

2.7 Conclusions

In this chapter, we have taken the first steps towards validating an existing model of mesh saliency through an eye-tracking-based user study. We have introduced the notion of normalized chance-adjusted saliency which is a robust measure of success of a mesh saliency model. We have observed significant correlations between the model of mesh saliency and human eye fixations. We believe that our carefully designed user study

can be useful for designing a better visual saliency model which is closer to human eye movements. This will also enable us to build further saliency-based systems for tasks such as visual enhancement.

Chapter 3

Persuading Visual Attention through Geometry

Artists, illustrators, photographers, and cinematographers have long used the principles of contrast and composition to guide visual attention. In this chapter we introduce geometry modification as a tool to persuasively direct visual attention. We build upon recent advances in mesh saliency to develop techniques to alter geometry to elicit greater visual attention. Eye-tracking-based user studies show that our approach successfully guides user attention in a statistically significant manner. Our approach operates directly on geometry, and therefore produces view-independent results that can be used with existing view-dependent techniques of visual persuasion.

3.1 Background

Artists have long used a rich collection of compositional and rendering techniques to persuade the viewers to pay greater attention to specific characters and objects in their paintings. The artist-determined visual hierarchy leads the viewer through the painting to see objects in the order of their relative importance, thereby finely controlling the communication of the message and purpose of their work. Image features such as luminance, color, and orientation are believed to guide the visual attention in the low-level human vision [90] and the role of luminance and texture contrast in attracting visual attention has been recently verified by Parkhurst and Niebur [92]. Recently, ideas inspired by the

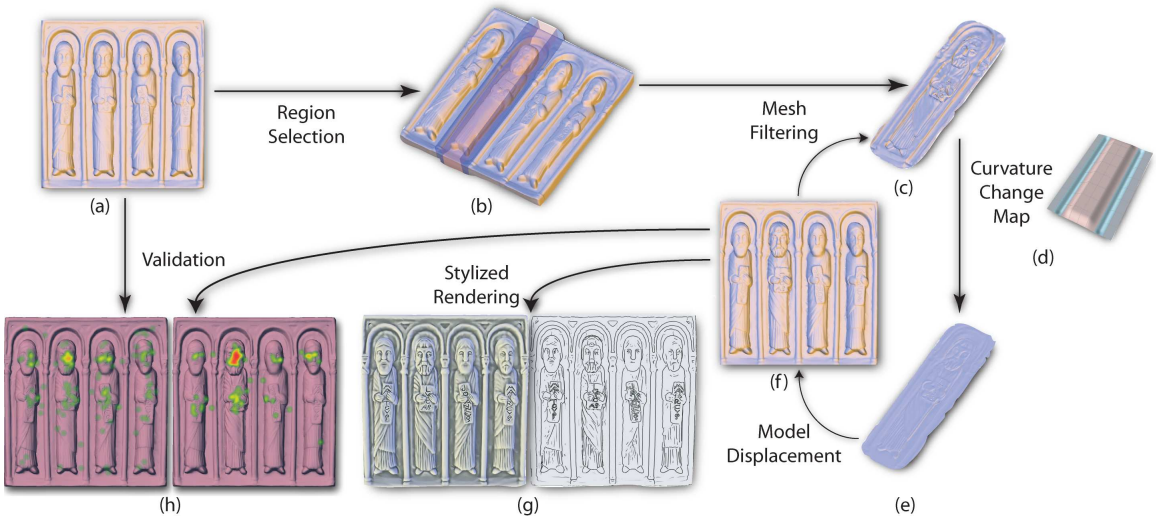


Figure 3.1: *An overview of persuading visual attention through geometry. The content creator defines a region of attention over the mesh in (a) as shown in (b); Mesh filtering over the desired attention region provides a set of displacements (c) along the vertex normals; Vertex displacements are weighted by a curvature change map (d) and then added to the input mesh. The resulting mesh in (f) elicits greater visual attention in the desired region (the second saint). Further, the mesh retains its visual attention persuasiveness through various rendering styles and illuminations (g) and this is validated by eye-tracking-based user study as shown in (h).*

principles of visual communication based in art, perceptual psychology, and cognitive science have begun to be carefully studied in the context of visual depiction [1]. As an example, discrepant lighting can be used to emphasize and depict user-specified detail in images [2] as well as meshes [68, 106].

Visual attention is a complex process that consists of an observer processing selected aspects of visual information more than others [90]. Eye movements are one of the most important, but not the only, means of visual selection. Many models of visual attention and saliency in an image have been evaluated by their ability to predict eye movements [91, 99]. Several computational models of visual saliency that model human attention have been developed. Based on Koch and Ullman's [66] model of early vision, Itti *et al.* [52] identify regions of images that are distinct from their neighbors by apply-

ing center-surround mechanisms to different multi-scale image feature maps and combine them to compute a saliency value for each pixel. There are a number of other approaches to modeling visual attention that combine local and global image features as in the work of Tsotsos *et al.* [122], Milanese *et al.* [84], Rosenholtz [105], and Torralba [120]. Santella *et al.* [108] have studied the relationship between eye movements and image saliency to explore the effect of local detail modulation to the way viewers examine an image. Enns and Rensink [30] have shown that salient pop-out phenomenon is not just limited to 2D image attributes but also occurs for 3D objects that differ in spatial orientation. Methods for computation of salient regions on 3D meshes have been developed by Watanabe and Belyaev [127], Hisada *et al.* [48], Lee *et al.* [69], and Howlett and O'Sullivan [49].

Saliency has been used in several 3D graphics applications including animation compression [81], rendering acceleration [135], viewpoint selection [69, 133], shape matching [36], and mesh simplification [49, 69]. Perception-based methods have been used for level-of-detail simplification for 3D meshes [77, 76]. Su *et al.* [114] have developed an elegant post-processing technique to reduce the salience of distracting regions in an image. They alter regional saliency by reducing its texture variation through the use of steerable pyramids [112]. In this dissertation we examine how saliency alteration in 3D geometry may affect viewer attention. We thereby aim to expand the scope of visual attention persuasion techniques from luminance contrast, color contrast, and texture contrast to include changes in geometry.

3.2 Overview

Visual attention can be guided in a goal-driven fashion or by external stimuli [98]. In this work we focus on the latter and therefore do not consider the high-level semantics of the objects or tasks at hand. Visual attention can be drawn to a specific region by simply having the selected pixels rapidly change and flash colors. Other approaches to draw attention to a region could include lighting it brightly, using high-saturation colors, or adding a high-curvature spike. Although these approaches would likely work, they should be considered coercive and obtrusive instead of being persuasive. The challenge in gently guiding visual attention is to do so in a finely nuanced style that only introduces subtle changes. In this dissertation we explore visual attention persuasion by making changes only to geometry. The various stages in our approach are shown in Figure 3.1 and summarized below.

Region Selection. We allow the content designer to specify attentional regions in a scene by directly selecting vertices, regions, or objects. The details of this interface are covered in Section 3.3.

Persuasion Filters. In Section 3.4 we discuss an approach to enhancing the saliency of a region to attract greater visual attention by designing a general class of mesh filters.

Validation. Once a persuasion filter has been applied to a given region we would like to have some objective evidence that it results in eliciting greater visual attention. Eye tracking is the most prevalent method of estimating visual attention. We have conducted an eye-tracking-based user study to verify the impact of our persuasion filters and report the results in Section 3.5.

Stylized Rendering. Our approach of geometry filtering incorporates visual attention persuasion early in the graphics pipeline. We have empirically observed that these changes are successfully propagated to the final rendering under several illumination and rendering styles. We discuss this in Section 3.5.4.

3.3 Region Selection

The input to our method is a mesh and a collection of one or more regions on it that have been selected by the user for eliciting greater visual attention. One of the simplest and most effective ways to specify regions on a mesh is by a WYSIWYG painting interface, such as the one developed by Hanrahan and Haeberli [44]. We use a similar screen-space brush in which the screen-space coordinates are projected back onto the surface by inverting the viewing transformation. This allows us to readily locate the appropriate mesh triangles and vertices in the vicinity of the brush location. The size of the brush is variable and can be modified to provide the user with coarse-to-fine control over the selected regions.

Figure 3.2: *A user can interactively define the regions on the mesh where greater attention is desired.*

Although initially it seemed attractive to allow a user to paint arbitrarily-shaped regions, we soon realized that this was not necessarily the best way to validate our results through user studies. The challenge here lay in attempting to isolate the effects of our persuasion filters from those arising naturally from the shape of the selected regions. Thus if a user specifies a *T*-shaped region on a mesh, the confluence of two linear regions is likely to elicit greater attention due to perceptual principles other than the ones we are targeting in this dissertation. Therefore we have only used simple circular or rectangular region selections. We have found it helpful to also add a spherical region selector. Figure 3.2 shows some of the regions selected by our region selection tool.

3.4 Persuasion Filters

Modifying the contrast of visually important features has been recently used in several contexts such as video abstraction [130] and attention-based modulation of detail for exaggerated shading [106]. Winnemoller *et al.* [130] regard the luminance and color opponency as visually important features and modified their contrast for image abstraction with the assumption that the changes in these frequencies are perceptually important.

We believe manipulating geometry by smoothing and sharpening is a basic but useful method for guiding viewer attention in a view-independent way for 3D graphics applications. However, applying this approach to a 3D mesh is not trivial and has several issues. First, we have to identify which 3D mesh property to modify by smoothing and sharpening operators just as luminance and color opponency are used in imagery. Second, we need a framework which can structure the modification of the geometry in a controlled

manner.

We change the mean curvature values at the vertices around the user-specified region by using bilateral displacements [33], and the amount of bilateral displacements is guided by the mesh saliency. In this section, we explain our design choices.

3.4.1 Bilateral Displacements

Taubin [119] introduced the Laplacian-operator-based isotropic mesh smoothing. An implicit formulation for isotropic mesh smoothing based on the analogy of geometric flow was developed by Desbrun *et al.* [25]. Anisotropic filtering techniques were subsequently developed by Guskov *et al.* [42], Desbrun *et al.* [26], Zhang and Fiume [136], Bajaj and Xu [6], and others. Bilateral filtering was introduced for meshes by Fleishman *et al.* [33] and Jones *et al.* [55] as a more stable alternative to anisotropic diffusion. These approaches are similar in formulation but differ in how they predict the local surface around a point.

There are a number of methods to alter the mean curvature values of a mesh. These include applying any kind of smoothing and sharpening operators explained above, displacing vertices, changing local normals, and even applying level-of-detail techniques. Amongst these, we would like to focus on methods that change the mean curvature values in a controlled manner.

Nealen *et al.* [85] captured high frequency details at a vertex $\mathbf{v_i}$ using the Laplacian

coordinates [25], [83] in their mesh editing framework as:

$$\mathbf{d_i} = \mathbf{v_i} - \frac{\sum_{\{i,j\}\in E} w_{ij}\mathbf{v_j}}{\sum_{\{i,j\}\in E} w_{ij}} \tag{3.1}$$

Here the vertices $\mathbf{v_j}$ are the one-ring neighbors of a vertex $\mathbf{v_i}$ and w_{ij} are determined using the cotangent weights [83]. Since $\mathbf{d_i}$ is in the local normal direction and the length $\| \mathbf{d_i} \|$ is proportional to the mean curvature around vertex $\mathbf{v_i}$, Laplacian-based vertex displacement offers a direct way to change mean curvature values in the selected region of interest. However, the vertex displacement $\mathbf{d_i}$ in equation (3.1) only operates for one-ring neighbors. We would like to guide the amount of displacements by mesh saliency. Since the mesh saliency operator does not operate at one-ring neighbors, but at a scale σ, we generalize the definition of neighbors with respect to the scale σ. Displacing vertices based on Fleishman *et al.*'s bilateral filter [33] satisfies this requirement while preserving all the benefits of Laplacian-based vertex displacement in changing mean curvatures. They smooth vertex $\mathbf{v}$ with a normal $\mathbf{n}$ as:

$$S(\mathbf{v}) = \mathbf{v} + d \cdot \mathbf{n} \tag{3.2}$$

$$d = \frac{\sum_{\mathbf{p}\in N(\mathbf{v},2\sigma_c)} W_c(\| \mathbf{v}-\mathbf{p} \|) W_f(< \mathbf{n}, \mathbf{v}-\mathbf{p} >) < \mathbf{n}, \mathbf{v}-\mathbf{p} >}{\sum_{\mathbf{p}\in N(\mathbf{v},2\sigma_c)} W_c(\| \mathbf{v}-\mathbf{p} \|) W_f(< \mathbf{n}, \mathbf{v}-\mathbf{p} >)}$$

Here $W_c(x) = e^{-x^2/2{\sigma_c}^2}$ is the closeness smoothing function with parameter σ_c that gives a greater weight to the vertices closer to the center vertex $\mathbf{v}$, $W_f(x) = e^{-x^2/2{\sigma_f}^2}$ is the feature weight function with parameter σ_f that penalizes a large variation in height from the local tangent plane, and $N(\mathbf{v}, 2\sigma_c)$ is the neighborhood of $\mathbf{v}$ containing all vertices $\mathbf{p}$ such that

$\| \mathbf{v} - \mathbf{p} \| < 2\sigma_c$. Note the similarities and differences between equations (3.1) and (3.2).

3.4.2 Saliency-guided Attention Persuasion

Lee *et al.* [69] have defined mesh saliency of a vertex $\mathbf{v}$ at a scale σ using the center-surround mechanism as:

$$\mathscr{S}(\mathbf{v}) = |G(\mathscr{C}, \mathbf{v}, \sigma) - G(\mathscr{C}, \mathbf{v}, 2\sigma)|$$

where $\mathscr{C}$ denotes the mean curvature values around a vertex $\mathbf{v}$ and $G(\mathscr{C}, \mathbf{v}, \sigma)$ is the Gaussian-weighted average of the mean curvature of vertices in the neighborhood $N(\mathbf{v}, 2\sigma)$. In fact, the center-surround mechanism used in this definition is the Difference of Gaussians (*DoG*) function at a fine scale σ and a coarse scale 2σ. We are interested in changing mean curvature values at the vertices around the user-specified region of interest so that those modifications result in user-desired saliency changes $\Delta\mathscr{S}$. For this purpose, we slightly modify the center-surround mechanism and define the center-surround operator at a vertex $\mathbf{v}$ using the Laplacian of the Gaussian-weighted averages as:

$$\mathscr{S}(\mathbf{v}) = w_1 G(\mathscr{C}, \mathbf{v}, \sigma) - w_2 G(\mathscr{C}, \mathbf{v}, 2\sigma)$$

where w_1 and w_2 indicate the positive weights of the Gaussian-weighted averages at a fine and a coarse scale, respectively. From this definition, the saliency change at a vertex $\mathbf{v}$

can be simply expressed as:

$$\Delta\mathscr{S}(\mathbf{v}) = w_1 G(\Delta\mathscr{C}, \mathbf{v}, \sigma) - w_2 G(\Delta\mathscr{C}, \mathbf{v}, 2\sigma)$$

where $\Delta\mathscr{C}$ is defined as the curvature change map. Given a user-specified saliency change map $\Delta\mathscr{S}$ we can compute the curvature change map ($\Delta\mathscr{C}$) around a vertex $\mathbf{v}$ by solving the following system of linear equations:

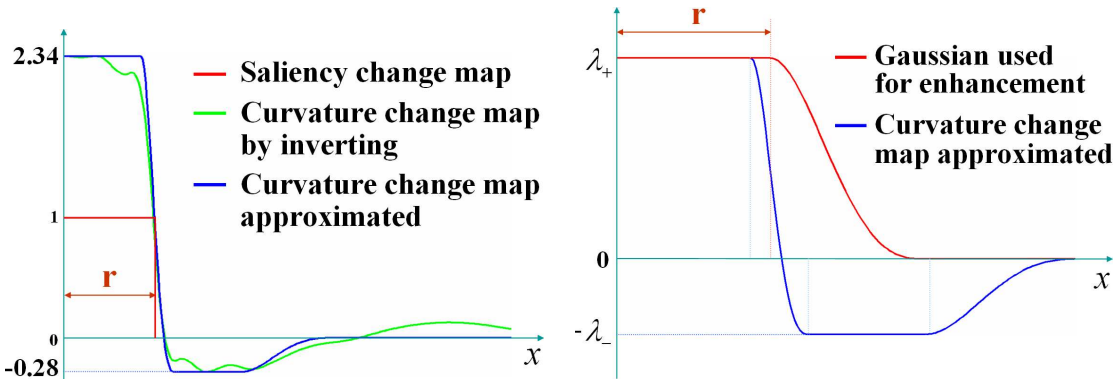
$$\begin{bmatrix} c_{1,1} & c_{1,2} & \dots & c_{1,n} \\ c_{2,1} & c_{2,2} & \dots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & \dots & c_{n,n} \end{bmatrix} \begin{bmatrix} \Delta\mathscr{C}(v_1) \\ \Delta\mathscr{C}(v_2) \\ \vdots \\ \Delta\mathscr{C}(v_n) \end{bmatrix} = \begin{bmatrix} \Delta\mathscr{S}(v_1) \\ \Delta\mathscr{S}(v_2) \\ \vdots \\ \Delta\mathscr{S}(v_n) \end{bmatrix}$$

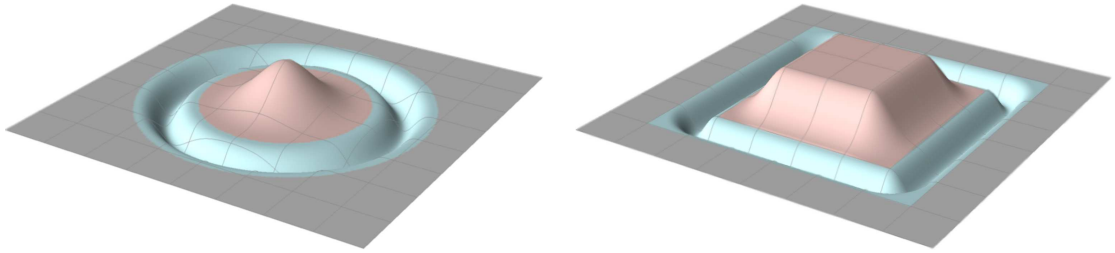
where the coefficients $c_{i,j}$ represent the difference between two Gaussian weights at scale σ and at scale 2σ for a vertex v_j in the neighborhood of the center vertex v_i. Setting the two weights w_1 and w_2 to be equal results in a rank-deficient system. We have observed that the system is stable for unequal weights. This corresponds to defining the saliency function using an aggregate of Difference-of-Gaussians (DoG) and a Gaussian (G) instead of just a DoG function. Specifically, we have found that using the weights $w_1 = 3/4$ and $w_2 = 1/4$, that corresponds to $1/4DoG + 1/2G$, results in a stable system that alleviates the rank-deficiency of the coefficient matrix.

We solve this system of linear equations at multiple scales σ_i to get the curvature change map ($\Delta\mathscr{C}_i$) at each scale σ_i. The overall curvature change map is computed as the multi-scale summation of $\Delta\mathscr{C}_i$. Assuming a spherical region of interest, the resulting

curvature change map has the shape of the green curve shown in Figure 3.3(a).

Since we are interested in emphasizing user-specified regions on meshes we generalize the spherical region of interest by using distance fields on a mesh [117]. Given a region of interest (*ROI*), we compute the distance field from the boundary of *ROI* and define the radius of the user-specified region r as the distance from the inner-most point to the boundary. Let $d_b(\mathbf{v})$ be the distance from the boundary of *ROI* to the vertex $\mathbf{v}$. We define $dist(\mathbf{v})$ as $r + d_b(\mathbf{v})$ for the vertex $\mathbf{v}$ which is outside of *ROI* and $r - d_b(\mathbf{v})$ for the vertex $\mathbf{v}$ which is inside of *ROI*. The resulting curvature change map based on





(c) (d)

Figure 3.3: *Saliency-guided curvature change map. Here r denotes the radius of the user-specified region. Figure (a) shows the curvature change map generated by multi-scale summation of curvature change maps in green and the approximation of it in blue with* $\lambda_+ = 2.34$ *and* $\lambda_- = 0.28$ *for comparison. Figure (b) shows the Gaussian that is used for enhancement for the user study in Section 3.5. Figure (c) and Figure (d) show the curvature change maps for circular and rectangular regions*

this distance field along the x-axis is shown for a rectangular region in Figure 3.1(d) and Figure 3.3(d).

We change the mean curvature values of the vertices in a mesh by using the bilateral displacements. We modify the mean curvature around a vertex $\mathbf{v}$ by displacing it as:

$$\mathscr{P}(\mathbf{v}) = \mathbf{v} - \Delta\mathscr{C}(dist(\mathbf{v})) \cdot d \cdot \mathbf{n} \tag{3.3}$$

where d is the displacement in the normal direction $\mathbf{n}$ for vertex $\mathbf{v}$ from the bilateral mesh filter in equation (3.2) and $\Delta\mathscr{C}(x)$ is the curvature change map. We refer to this as the *persuasion filter* $\mathscr{P}$.

In practice, we approximate the computed curvature change map by piecewise C^2-continuous degree 4 polynomial radial functions inspired by [129] as:

$$\Delta\mathscr{C}(x) = d(x) + w(x) \cdot g(\rho(x), \sigma(x))$$

$$g(\rho, \sigma) = \begin{cases} (1 - \frac{\rho}{2\sigma})^3(\frac{3\rho}{2\sigma} + 1) & \text{,if } \rho \leq 2\sigma \\ 0 & \text{,otherwise} \end{cases}$$

where $d(x)$, $w(x)$, $\rho(x)$,and $\sigma(x)$ are determined by:

$$\begin{cases} d = \lambda_+, w = 0 & \text{,if } x < \frac{7}{8}r \\ d = -\lambda_-, w = \lambda_+ + \lambda_-, \rho = x - \frac{7}{8}r, \sigma = \frac{3}{16}r & \text{,elif } x < \frac{10}{8}r \\ d = -\lambda_-, w = 0 & \text{,elif } x < 2r \\ d = 0, w = -\lambda_-, \rho = x - 2r, \sigma = \frac{1}{2}r & \text{,elif } x < 3r \\ d = 0, w = 0 & \text{,otherwise} \end{cases}$$

Our approximating function is shown by the blue curve in Figure 3.3(a). In our current implementation we use $0.1 \leq \lambda_- \leq \lambda_+ \leq 0.3$. For all the examples in this dissertation we have applied the persuasion filter five times to the region of interest.

Figure 3.3(b) shows the Gaussian fall-off function used for the comparisons in Section 3.5. The Gaussian fall-off function has been used in regional enhancement for volume illustration [104], in attention-based modulation of detail for exaggerated shading [106],

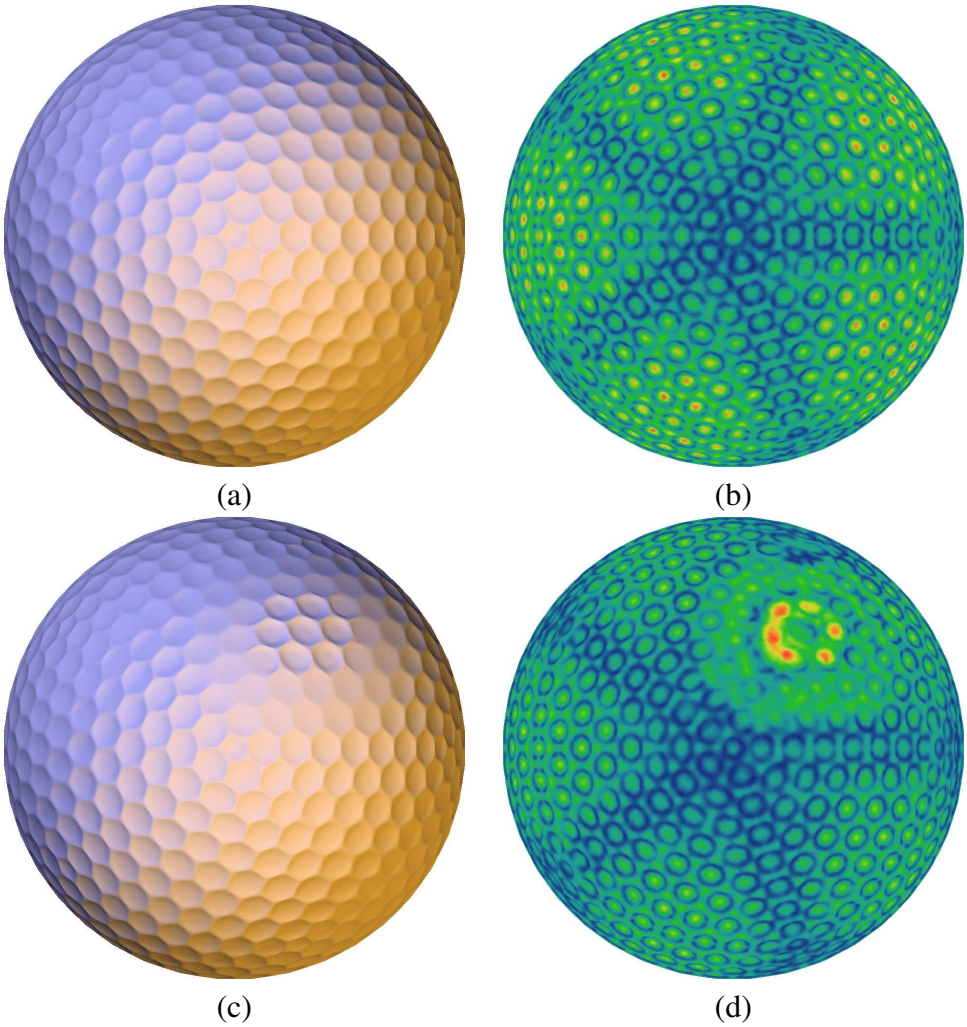


Figure 3.4: *Images show the saliency changes before and after the application of the persuasion filter on the Golf ball model. Image (a) shows the original model and Image (b) shows its original saliency. Image (c) shows the result of applying the persuasion filters to the top part of the Golf ball with* $(\lambda_+ = 0.2, \lambda_- = 0.2)$*, and Image (d) shows its saliency.*

and in stylized rendering [18]. There are several ways to define the Gaussian fall-off function. In this dissertation, we have used the Gaussian fall-off from the boundary of the region of interest instead of from a point to more closely match our computed curvature change map. We call the filter based on this Gaussian fall-off function as the *Gaussian filter*. For our evaluations we have kept the maximum change (λ_+) of the Gaussian filter to be the same as the maximum change of the approximated curvature change map in blue in Figure 3.3(b) and the standard-deviation (σ) of this function from the boundary of the region of interest is determined to make the area under its curve equal to that under the persuasion filter curve (to equalize the changes introduced by the two filters).

Figure 3.4 shows the application of the persuasion filter around the front part of the Golf ball model that results in an average saliency increase of 101.6% for the region.

Figure 3.7 further shows the enhancement by persuasion filter in the last row. Compared to the original models in the first row, the effects of applying our filters are clearly visible. The application of the persuasion filter depends on the number of vertices in the region of interest. For the lower left part of the Golf ball model in Figure 3.7, 11.2K vertices were involved and it took 0.8 seconds. For one of saints in Figure 3.7, 26K vertices were involved and it took 2 seconds. When one of the Armadillos in Figure 3.7 was selected, there were 650K vertices involved in the persuasion filtering and it took about 15 seconds. These times were measured on a 2.93 GHz Intel(R) Core(TM)2 CPU Windows PC with 4 GB RAM.

3.5 Validation and Results

As we discussed in the previous section, the geometry-based model of visual attention persuasion results in an increase in mesh saliency over the user-specified region. The real measure of success however is whether this actually results in eliciting greater viewer attention. To gather objective evidence of the effectiveness of our approach we have carried out an eye-tracking-based user study and have analyzed the results in two different ways: percentage of fixation points on the region of interest and average duration of consecutive fixation points on the region of interest. The first analysis is to see if our technique attracts the viewers' gaze to the region of interest while the second analysis reveals if it holds the viewers' gaze. An excellent guide to the experimental design of eye-tracking-based user studies appears in Parkhurst *et al.* [91].

3.5.1 Hypotheses

There are two hypotheses for each analysis, resulting in a total of four hypotheses. Our first hypothesis (H1) is that the eye fixations increase over the region of interest by using the persuasion filter as compared to the original, unaltered model. Our second hypothesis (H2) is that the eye fixations increase over the region of interest by using the persuasion filter compared to the use of a Gaussian filter. Our third hypothesis (H3) is that the average durations of eye fixations on the region of interest increase by using the persuasion filter as compared to the original, unaltered model. Our fourth hypothesis (H4) is that the average durations of eye fixations on the region of interest increase by using the persuasion filter compared to a Gaussian filter. We next examine the validity of these

hypotheses. We examine hypotheses H1 and H2 in Section 3.5.3.1 and hypotheses H3 and H4 in Section 3.5.3.2.

3.5.2 Experimental Design

Data were collected from a total of 18 subjects participating for pay. They had normal or corrected-to-normal vision and were not familiar with the goals of this study. Subjects were told to freely view the images with no assigned goal. General settings of eye-tracker and its calibration are similar to those explained in Section 2.3.

Duration: The user study had 13 trials (images) including 4 irrelevant images. Each trial started with the subject seeing a blank screen with a cross at the center of the screen. The subject was asked to look at the cross before clicking the mouse to bring up the next image. This ensured that each trial started with the subject's eyes fixated at the center of the image. Each image was shown for 5 seconds. Each study took about 80 seconds. Two irrelevant images were shown at the start of the experiment to give the subject a sense of the duration.

Image Ordering: There were a total of 15 images used over all the experiments. Each image set consists of one original image and four filtered images in which one of the two regions is enhanced by either a Gaussian or our persuasion filter. Images and the regions of interest were carefully chosen so that two regions of interest in each image set have similar shape, size, and saliency values (as computed by [69]). We have used the Golf ball, the Romanesque Relief, and the Armadillos models shown in Figure 3.7 for our study. Each user saw 9 images out of these 15 images. When we ordered the im-

ages for each user, we considered differential carryover effects and the counter-balancing problem. First, we placed similar images far apart to alleviate differential carryover. At the same time, we did not place the similar images in perfectly regular manner so that a user could not predict the next image. We did this by inserting a couple of unrelated images. Alleviating differential carryover effect had the highest priority in our ordering because each user looked at 3 similar images (original and filtered with two different techniques). Finally, we randomized the order of regions and the order of enhancement types (Gaussian and Persuasion-filtered) to counterbalance overall effects.

Fixation Points: We divide the raw data points into fixation points and saccade points as described in Section 2.3.3. After we removed short fixations, we weighted the contribution of each fixation point by its duration to give more weight to longer fixations. For the rest of this chapter, when we refer to a fixation point we imply a duration-weighted fixation point. For every image that we used in our study, we had a specified region of desired visual attention.

3.5.3 Data Analysis

3.5.3.1 Percentage of Fixation Points

The results of our study can be seen in Figure 3.5 and they show the increase in fixation points on the regions selected by the user. Each grouping of bars in Figure 3.5 is labeled by the object or region on which the filters was applied. Figure 3.6 shows the increase in the number of fixation points on the attention area after the application of persuasion filters. In Figure 3.6(b), the third saint was processed with persuasion filters

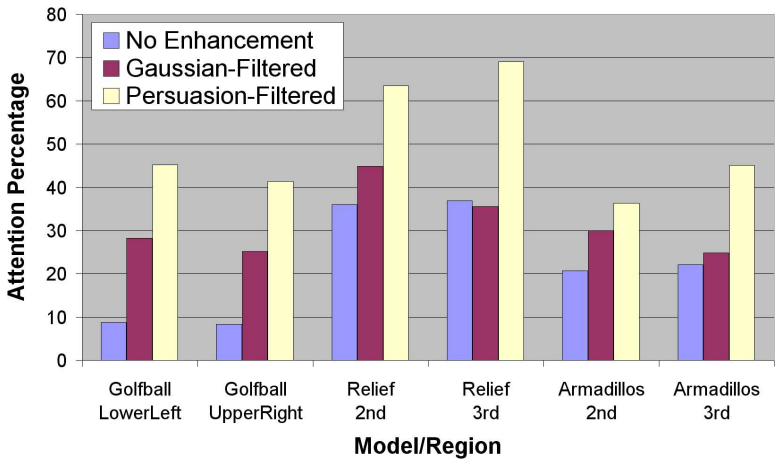
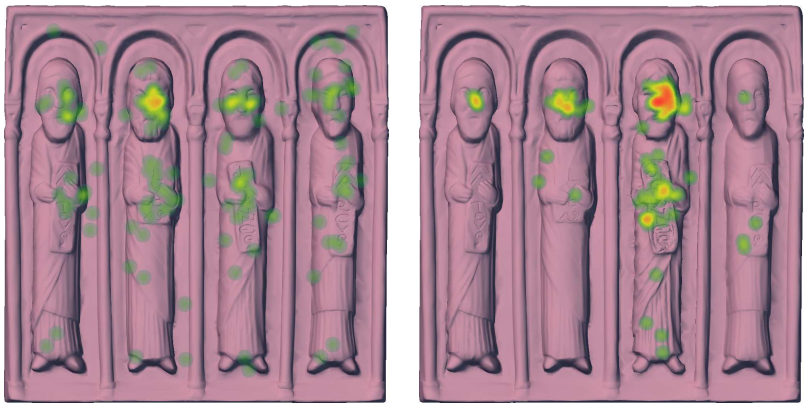


Figure 3.5: *The bars show the average percentage of fixation points on the region of interest for the original, the Gaussian-filtered and the Persuasion-filtered models.*



(a) (b)

Figure 3.6: *Image(a) and (b) show the fixation points on the original model and the model altered by persuasion filter, respectively. Fixation points are recorded over the first* 5 *seconds from* 9 *subjects, and visualized with hot spot map where warm colors show the areas of highest fixation count.*

Table 3.1: *The ANOVA Tests*

Condition	F-Value	p-Value
Enhancement techniques on three sets	30.09	< 0.001
Enhancement techniques on Golfball	13.45	< 0.001
Enhancement techniques on Relief	9.728	< 0.001
Enhancement techniques on Armadillos	7.2	0.002
Gaussian vs. Persuasion on three sets	21.23	< 0.001

and this resulted in 87% increase in percentage of fixation points on it.

Table 3.2: *The Pairwise t-tests (two-tailed)*

Model	Condition	t-Value	p-Value
Golfball	No Change vs. Gaussian	-3.34	0.004
	No Change vs. Persuasion	-5.84	< 0.001
	Gaussian vs. Persuasion	-2.81	0.012
Relief	No Change vs. Gaussian	-0.47	0.647
	No Change vs. Persuasion	-4.69	< 0.001
	Gaussian vs. Persuasion	-3.95	0.001
Armadillos	No Change vs. Gaussian	-1.56	0.138
	No Change vs. Persuasion	-3.42	0.003
	Gaussian vs. Persuasion	-2.31	0.033

We have carried out a two-way ANOVA on the percentage of fixations for two conditions: filtering methods and image sets. As shown in Figure 3.5, participants fixated more on the regions of interest when they are filtered with enhancement techniques. Overall, there was a strong difference for the percentage of fixations depending on enhancement techniques, $F(2,153) = 30.09, p < 0.001$.

Since we filtered two regions of interest for each image set, we did another analysis with filtering methods and regions for each image set. Amongst regions, there were no statistically significant differences ($F(2,48) = 0.027 \sim 0.195, p > 0.661$) as expected. For enhancement methods, there were statistically significant differences for all the models. We also carried out a two-way ANOVA test with two enhancement methods (Gaussian and Persuasion) to see if the Persuasion-based enhancement is better. The result shows a significant improvement on the percentage of fixations with Persuasion-based enhancement, $F(1,102) = 21.23, p < 0.001$.

Next, we performed a pairwise t-test on the percentage of fixations before and after we apply the enhancement techniques for each model (this is the only condition in the

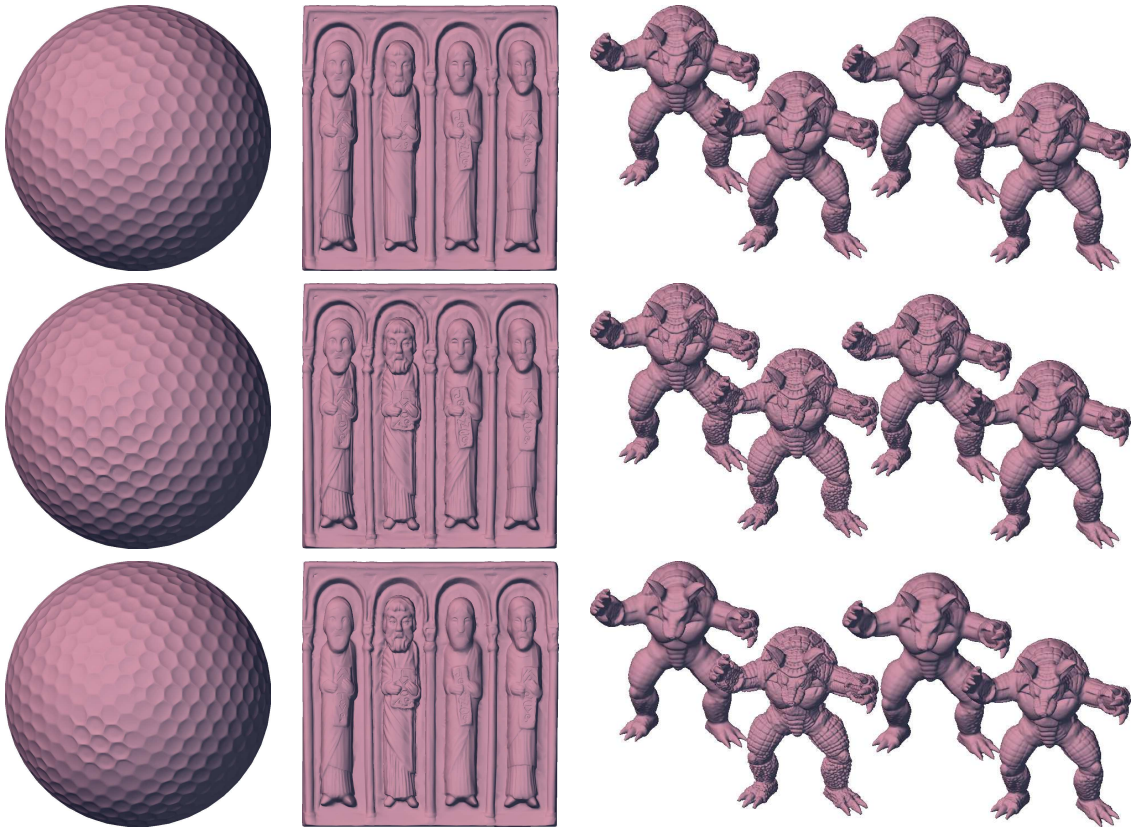


Figure 3.7: *A subset of images used in the user study. The first row shows the original models. The second row shows enhancement by the Gaussian filter. The last row shows the enhancement by the persuasion filter. Filters were applied to the lower left part of the Golfball model with* $(\lambda_+ = 0.2, \lambda_- = 0.2)$*, the second saint of the Romanesque Relief model with* $(\lambda_+ = 0.2, \lambda_- = 0.1)$*, and the second Armadillo of the Armadillos model with* $(\lambda_+ = 0.3, \lambda_- = 0.3)$*, respectively.*

test). Table 3.2 shows the results for each of the models. We found a significant difference in the percentage of desired fixations after we applied the persuasion filters for each of the models. There was a significant difference in the percentage of desired fixations for the Gaussian filters only for the Golf ball model. When we carried out a pairwise t-test between two filtering methods (Gaussian and Persuasion) for each image set, we still observed a significant difference for each case.

The results in this section clearly validate that there are significant increase of fixations on the regions of interest by the persuasion filter over the original as well as the

Gaussian-filtered meshes.

3.5.3.2 Average Duration of Consecutive Fixation Points

Towards the end of Section 3.5.2 we discussed why we weighted the contribution of each fixation point by its duration. However, simply weighting fixation points by their duration cannot distinguish frequent brief fixations from single long fixation. Henderson *et al.* [47] advocate that fixation points correlate with attention when they are longer in duration. To show that the images enhanced by our persuasion filter actually hold viewer gaze, we have done another analysis on the average duration of fixation points on regions of interest. Table 3.3 shows the result of the average duration and their standard deviation of fixation points across the subjects for each model in seconds.

Since the standard deviations are large, we assessed the statistical significance of these results through ANOVA. We carried out a two-way ANOVA on the average durations for two conditions: enhancement methods and regions. Each image set has two regions of interest for a data set (Golfball, Relief and Armadillos). Overall, there

Table 3.3: *The Average Duration and Standard Deviation of Fixation Points across the Subjects in seconds*

Model	Condition	Average	Std Dev.
Golfball	No Change	0.204	0.270
	Gaussian Enhancement	0.374	0.351
	Persuasion Enhancement	0.744	0.467
Relief	No Change	0.483	0.541
	Gaussian Enhancement	0.457	0.269
	Persuasion Enhancement	1.000	0.631
Armadillos	No Change	0.408	0.318
	Gaussian Enhancement	0.476	0.354
	Persuasion Enhancement	0.678	0.618

was a strong difference for the average durations depending on enhancement techniques, $F(2,153) = 15.37, p < 0.001$. Amongst regions, there were no statistically significant differences ($F(2,48) = 0.49\ 2.735, p > 0.105$), as expected.

We carried out a two-way ANOVA test comparing the unaltered (original) and Persuasion-enhanced meshes with the two regions. The result shows a significant improvement on the average durations with Persuasion-based enhancement, $F(1,102) = 21.65, p < 0.001$. We also carried out a two-way ANOVA test with two enhancement methods (Gaussian and Persuasion) with the two regions. The result shows a significant improvement on the average durations with Persuasion-based enhancement, $F(1,102) = 16.96, p < 0.001$.

The results validate the statistically significant increase of average durations on the regions of interest by the persuasion filter over the original as well as the Gaussian-based methods.

3.5.4 Stylized Rendering

We have empirically tested our models with and without the application of the persuasion filters with several illumination and rendering styles. These include the standard OpenGL lighting model and suggestive contours [23]. The results of the application of our persuasion filters are clearly discernible with each of these lighting models and illumination styles. These are shown in Figure 3.8. The reason for the successful propagation of fine geometry alterations to the final rendered image could be that the bilateral displacements allow us to preserve and enhance edges in the target attention area, while smoothing

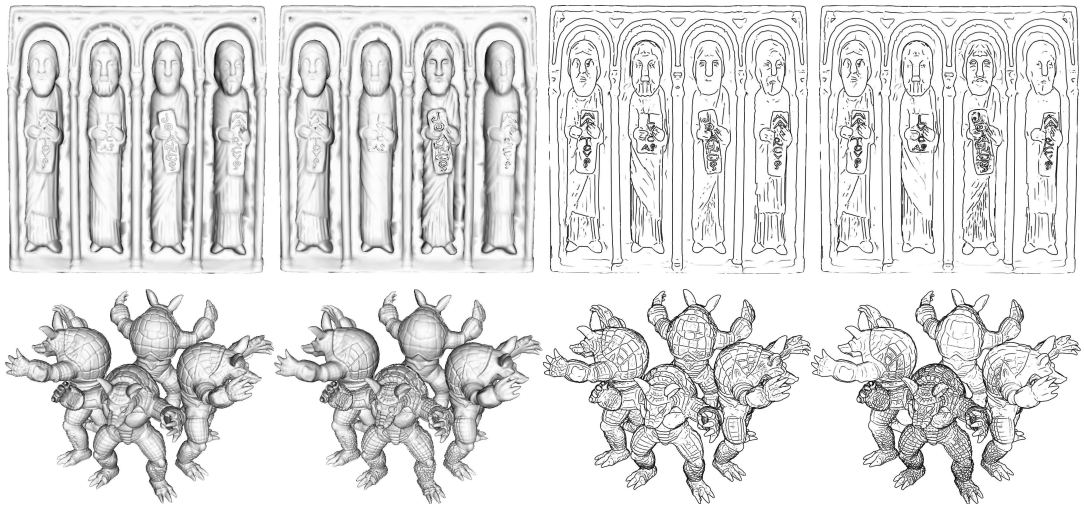


Figure 3.8: *Illustration of the Romanesque Relief and the Armadillos by Lambertian lighting and Suggestive Contours. The first and the third column show the original models while the second and the fourth column have persuasion filters applied to the third saint with* $(\lambda_+ = 0.2, \lambda_- = 0.1)$ *and to the frontmost Armadillo with* $(\lambda_+ = 0.3, \lambda_- = 0.3)$*, respectively.*

them around it. These effects are perhaps most clearly visible in the suggestive contour rendering emphasizing the third saint and the closest Armadillo in Figure 3.8.

Since persuasion filters are applied to the geometry before the graphics pipeline, this process is view-independent. The empirical results of this section suggest that persuasion filters may also be able to elicit attention across a wide class of illumination and stylized rendering techniques. This encourages us to believe that it may be possible to combine geometry-based visual attention persuasion techniques with other visual attention persuasion techniques that are independent of geometry, such as luminance contrast, color contrast, and texture contrast.

3.5.5 Discussion

We believe that this work presents an exciting new direction in visual attention persuasion through geometry. At the same time it is important to note the underlying assumptions, limitations, and distinctions from other related work.

In this dissertation, we have used bilateral displacements for smoothing and sharpening operations to alter the frequencies in the geometry. Therefore, if a surface is completely flat and its mean curvature is zero at every vertex, there is no enhancement by our persuasion filter since the displacement of the bilateral filter would be also zero. Also, we should note that sharpening is an inherently unstable process. We have tried to reduce these effects by using small delta values ($0.1 \sim 0.35$) iteratively (5 times) in our dissertation. However, repeated application of our persuasion filter can eventually produce degenerate triangles. We are planning to incorporate remeshing [116, 125] and subdivision operators [29, 138] to incrementally remove degenerate triangles produced during the repeated application of our persuasion filter in the future.

Our desired saliency change map contains content at all frequencies but our saliency operator, which has become a convolution, is band-pass. As a result, we use unequal weights w_1 and w_2 to reproduce all the frequencies of our saliency change map. Also, when we invert the saliency computation, we have only considered positive weights because we are interested in finding one assignment of the curvature change map which satisfies the saliency computation. Positive weights w_1 and w_2 emphasize the center and de-emphasize the surrounding while negative weights achieve the opposite. We believe that there could be an application which can get benefit from de-emphasizing the cen-

ter and emphasizing the surrounding, and plan to study the implications of the negative weights in the future.

Our technique is mainly designed to increase mesh saliency values around the selected region of interest. It could sometimes change the most salient regions in a mesh. The overall strength of such a change with respect to other salient features should be studied further. Also, we should note that even though there is a qualitative saliency enhancement for the desired region as in Figure 3.4, we have not quantified the change in presence of other salient regions on the mesh.

In this chapter we have provided a technique to guide viewer attention by only altering the geometry of the mesh and not its other appearance attributes. We did this because we wished to systematically establish the role of geometry alteration in persuading visual attention. Our method directly operates on meshes and therefore is more general than prior art that only operates at the level of rendering stylization. Further our method is view-independent and therefore by pushing the influence of attention deeper into the graphics pipeline, allows content creators to have greater flexibility in using other conventional techniques later in the graphics pipeline. Our initial results in this direction have raised a number of other interesting questions and issues that we hope subsequent research will be able to address. For instance, we do not yet know if geometric techniques for persuading visual attention will prove to be more or less powerful than other visual attention persuasion techniques based on say, luminance, chrominance, or texture contrast. How do these multiple channels of visual attention persuasion interact with each other's effectiveness – do they have mutual reinforcements or cancellations?

An important assumption for us has been to consider visual attention persuasion in

a task-independent setting. This is similar in spirit to the early work in image saliency that relied on determining what constitutes the "pop-out" phenomenon during the highly-parallel pre-attentive state of the human visual system. We have attempted to limit the role of semantics in three ways: (i) we presented each image for only 5 seconds, (ii) we did not give any tasks to the users when viewing images, and (iii) we had users view 3D models that were composed of semantically similar objects such as multiple Armadillos and multiple saints. We hope that once geometric visual attention persuasion has been firmly established in a pre-attentive task-independent setting, it will lead to further work on semantically-based techniques for persuading visual attention using geometry alteration.

Previous work on user-guided simplifications [59, 72] and spatially-varying detail control [86] enables detail alteration based on user preferences or view dependence for rendering acceleration and for reducing visual clutter. Our work differs in its underlying motivation as well as approach in that we develop and validate a mechanism to guide viewer attention through geometry alteration.

Finally, we would like to mention that although our filter as seen in Figure 3.3 may appear to be similar, at a first glance, to the Difference-of-Gaussians (DoG) filter, it is not; the classical DoG does not have a flat top.

3.6 Conclusions & Future Work

Visual attention persuasion can be helpful in 3D graphics in several contexts – visually guiding users through complex graphics, facilitating interactions with attention-aware

graphics tools and applications, and providing users with a more rewarding experience by guiding their attention to regions and objects desired by content creators. We have defined persuasion filters for meshes by inverting the center-surround saliency operator. Our user study shows that persuasion filters are able to draw greater user attention and that this condition is statistically significant. Our approach adds geometry alteration to the list of techniques at the disposal of visual persuaders.

At present our filters are not informed by the local mesh properties. Kim and Rossignac [61] have recently developed a mesh filtering framework that can compute filter parameters based on user-specifiable mesh features. This is an interesting development that could lead to design of persuasion filters that could be guided not just by regions but also by features. Another interesting direction to explore will be to incorporate view-dependence in persuasion filtering. As mentioned earlier, in this work we have exclusively considered the role of geometry in the context of visual attention persuasion. It will be interesting to see how other visual attention persuasion channels of color, luminance, and texture contrast interact with geometry alteration using the techniques of persuasion filters presented here.

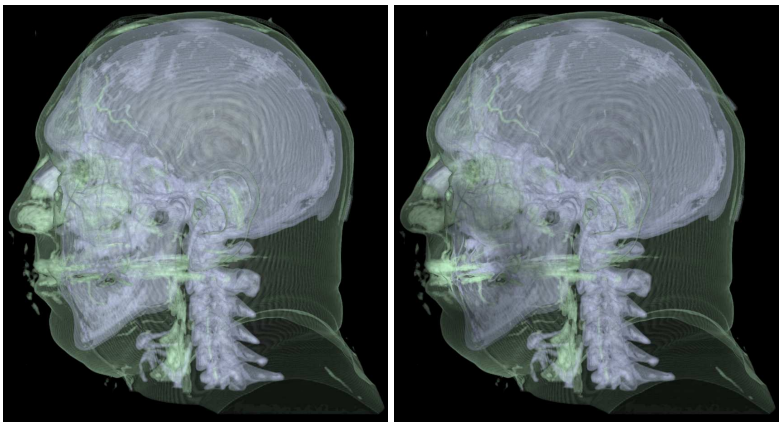
Chapter 4

Saliency-guided Enhancement for Volume Visualization

In previous chapter we introduced saliency-based visual enhancement for 3D meshes through geometry modification. In this chapter we present a visual-saliency-based operator to enhance selected regions of a volume. We show how we use such an operator on a user-specified saliency field to compute an emphasis field. We further discuss how the emphasis field can be integrated into the visualization pipeline through its modifications of regional luminance and chrominance. Finally, we validate our work using an eye-tracking-based user study and show that our new saliency enhancement operator is more effective at eliciting viewer attention than the traditional Gaussian enhancement operator.

4.1 Related Work

Direct volume rendering models the attenuation of light in a volume composed of particles with varying densities and opacities [58, 70]. Volume rendering has evolved considerably over the past two decades and now engineers, scientists, medical researchers, and visual designers use a rich suite of tools and techniques to specify the visual appearance of a volume based on their needs. Transfer functions have played a crucial role in broad use of direct volume rendering. The design of transfer functions to generate informative visualizations has been a significant challenge that has been addressed by a number of researchers [95]. A number of heuristics are used to guide the users in selecting



(a) (b)

Figure 4.1: *Saliency-guided Enhancement for Volume Visualization: Image (a) shows the traditional volume visualization and image (b) shows the result of applying our saliency-guided enhancement operator to the mouth.*

appropriate transfer functions. For instance, Levoy [70] suggested the use of the gradient magnitude to identify surfaces in volume data. Kindlmann and Durkin [62] used the first and second derivatives along the gradient direction to calculate a boundary emphasis to be included in the opacity transfer function. In addition to the design of the opacity transfer function, general multi-dimensional transfer functions were studied to better convey the boundaries and features in volume data [63, 64, 65, 78].

Stylized rendering in volume visualization has attracted extensive research interest in the last few years. Treavett and Chen [121] developed techniques for pen-and-ink illustrations of surfaces within volumes. Lu *et al.* [75] used stippling techniques for interactively previewing large datasets. Burns *et al.* [15] identified and depicted silhouettes and suggestive contours in volumes. Rheingans and Ebert [104] developed a variety of volume illustration techniques. They used the scalar field gradient in addition to the local density to carry out a sophisticated set of perceptual enhancements that are view- and light-dependent. Interrante *et al.* explored the use of carefully oriented textures to convey

surface shape in volumes [50] and to convey 3D flows [51]. With few exceptions, all of these enhance the important features based on the volume sample value or the local volume characteristics.

Viewers pay greater visual attention to regions that they find salient [90]. Therefore, many models of visual attention and saliency in an image have been evaluated by their ability to predict eye movements [91, 99]. Conversely, eye movements have been used to guide meaningful abstractions of photographs [24, 108] and volume composition [74]. Several computational models of visual saliency that model human attention have been developed. Itti *et al.* [52] developed a computational model of visual attention based on the center-surround operators in an image. Recently, Lee *et al.* [69] have proposed saliency for meshes based on a multi-scale center-surround mechanism that operates on local curvature.

Once saliency for a volume is computed either by using eye-tracking data, or through computational models of human perception, or through feature extraction, it can be used to better inform the visualization process. Machiraju *et al.* [79] used feature-based saliency to perform progressive visualization. They first project the volume data into a wavelet basis and identify features at multiple scales. Then, they use the ranked regions of interest in a priority scheduling scheme to progressively visualize the data. Rheingans and Ebert [104] suggested the idea of importance-based regional enhancement for volume illustration. Their approach involves enhancing a region around a user-specified point of interest using a gradual fall-off function based on the view direction. Viola *et al.* [124] developed an innovative importance-driven approach to emphasize features in volumes. Their approach modulates the opacity of a feature based upon its importance as well as

the importance of the features that it occludes. This approach has been shown to be very valuable in simultaneously visualizing interior and exterior structures of a volume in clutter-free renderings that show the important regions while suppressing or eliminating the less important regions [13]. Hauser [45] suggested emphasizing regions in volumes using opacity, color, frequency (focus), and rendering styles.

A very interesting beginning in altering saliency to guide viewer attention has been made by Su *et al.* [114]. They have developed an elegant post-processing technique to reduce the salience of distracting regions in an image. They alter regional saliency by reducing its texture variation through the use of steerable pyramids and validate their results with eye-tracking-based user studies.

In this dissertation we propose a new enhancement operator for emphasizing regions of volumes. Our enhancement operators are based on the idea of reversing the visual saliency computation at multiple scales and we show that they can be used to guide viewer attention. We integrate the application of our enhancement operator to the visualization pipeline through an emphasis field that could be used to modulate luminance and chrominance to enhance visual perception of volume data.

4.2 Overview

Guiding user attention in volume visualization is an important component of the overall visual experience. Visual attention can be achieved by obtrusive methods such as very bright or flashing pixels in the desired region. However, such techniques distract the viewer from adequately exploring other regions of the volume data. Artists and

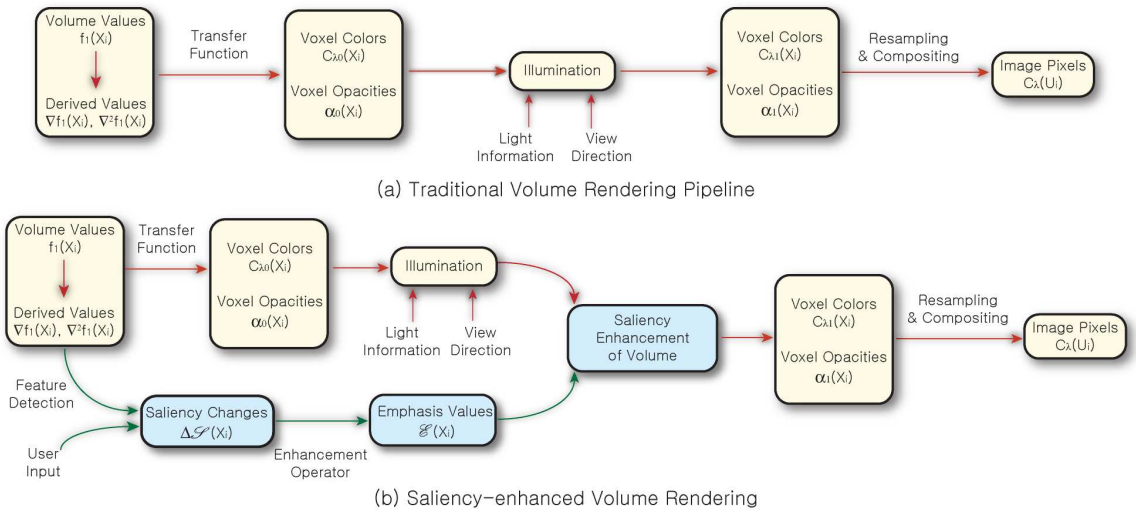


Figure 4.2: *(a) The traditional visualization pipeline. (b) Saliency-enhanced visualization pipeline. The saliency field is modified by the enhancement operator to generate an emphasis field. The emphasis field is used to enhance the perception of features in volume by modulating appearance attributes such as luminance, chrominance, and texture detail.*

illustrators have long used the principles of visual perception to gently guide viewer's attention to regions and objects that they wished to emphasize. In the preceding section we have provided a summary of several techniques used in volume visualization to emphasize regions. Most emphasis methods involve increasing the perceptual importance of a given region through a Gaussian fall-off function centered at the region of interest. Such Gaussian functions have been used to modulate opacity, luminance, chrominance, and texture detail. In this dissertation we present a novel saliency-guided enhancement operator based on computational models of visual saliency and show that it is better at drawing visual attention than a Gaussian. The various stages in our approach are shown in Figure 4.2 and summarized below.

Saliency Field. We assume that a saliency change is assigned to each voxel of the volume data. This assignment could be based upon user specification (manual painting),

eye-tracking data, or feature computation.

Enhancement Operator. We introduce a general class of saliency-guided enhancement operators that generate an emphasis field from a saliency field. These operators are based on the center-surround mechanisms at multiple scales and invert the process of the saliency computation at each scale.

Emphasis Field. Emphasis field is used to guide the modulation of the visual appearance by locally changing luminance or chrominance. Note that since the emphasis field operates independently of the transfer function, its effects on the overall visualization pipeline are complementary to those achieved by transfer functions alone.

Validation. We would like to have some objective evidence that our saliency-based enhancement operators elicit greater visual attention than the original volume visualization as well as the traditional Gaussian-based enhancement. We have conducted an eye-tracking-based user study to verify the impact of our enhancement operators and report the results of our method in Section 4.5.

4.3 Emphasis field computation

The starting point for our approach is the generation of a saliency field $\Delta\mathscr{S}$ that defines a desired saliency change for every voxel. We assume that the saliency field for each voxel defines its importance on a scale from 0 to 1. Such a saliency field could be specified through a number of methods. The first possibility is to acquire it by recording a user's eye movements when a given volume is shown [74]. Another possibility is for an illustrator or a domain expert to specify the desired saliency changes for one or more

voxels [45, 104, 124]. A third possibility is to procedurally detect and rank features in the order of their importance [79].

We would like to define an enhancement operator guided by the saliency field that is used to increase the perceptual importance. For this we start with a computational model of saliency. Itti *et al.* [52] have defined saliency using the center-surround mechanism on the non-oriented properties such as intensity and color in an image at multiple scales. Lee *et al.* [69] have recently defined mesh saliency using the center-surround mechanism on the mean curvature at each vertex at multiple scales. Since the overall volumetric appearance is a multivariate process, we use the above idea to compute the saliency field on a virtual emphasis field $\mathscr{E}$. The emphasis field can then be used to logically decouple the processes of specifying multi-scale enhancement and achieving it through modulation of various volumetric appearance such as color and opacity. Let a voxel v_i be the i-th voxel within a volume V. Then, let $\Delta\mathscr{S}(v_i)$ and $\mathscr{E}(v_i)$ be the saliency change and the emphasis value for a voxel v_i, respectively. We define the saliency change for a voxel v_i using the center-surround mechanism L of the emphasis field $\mathscr{E}$ at scale, σ as:

$$\Delta\mathscr{S}(v_i) = L(\mathscr{E}, v_i, \sigma) \tag{4.1}$$

In general, there can be infinitely many solutions for an emphasis field $\mathscr{E}$ that will give us a desired value of saliency field $\Delta\mathscr{S}$ depending on the definition of the center-surround operator L. Let $G(\mathscr{E}, v_i, \sigma)$ be the Gaussian-weighted average of the emphasis field centered at a voxel v_i:

$$G(\mathscr{E}, v_i, \sigma) = \sum_{v_j \in V} \mathscr{E}(v_j) g(v_i, v_j, \sigma) \tag{4.2}$$

where $g(v_i, v_j, \sigma) = \frac{exp[-\|v_j - v_i\|^2/(2\sigma^2)]}{\sum_{v_k \in V} exp[-\|v_k - v_i\|^2/(2\sigma^2)]}$.

We define the center-surround operator at a voxel v_i using the Laplacian of the Gaussian-weighted averages as:

$$L(\mathscr{E}, v_i, \sigma) = w_1 G(\mathscr{E}, v_i, \sigma) - w_2 G(\mathscr{E}, v_i, 2\sigma) \tag{4.3}$$

where w_1 and w_2 indicate the weights of the Gaussian-weighted averages at a fine and a coarse scale, respectively. Positive weights w_1 and w_2 emphasize the center and de-emphasize the surrounding while negative weights achieve the opposite. In our work, we have used positive weights.

4.3.1 Saliency-Enhancement Operator

Let us reformulate the saliency change for a voxel v_i using Equations 4.1- 4.3:

$$\begin{aligned} \Delta\mathscr{S}(v_i) &= w_1 \sum_{v_j \in V} \mathscr{E}(v_j) g(v_i, v_j, \sigma) - w_2 \sum_{v_j \in V} \mathscr{E}(v_j) g(v_i, v_j, 2\sigma) \\ &= \sum_{v_j \in V} \mathscr{E}(v_j) \cdot c_{i,j} \end{aligned}$$

where $c_{i,j} = w_1 g(v_i, v_j, \sigma) - w_2 g(v_i, v_j, 2\sigma)$. We can express the above as the following system of simultaneous linear equations:

$$\begin{bmatrix} c_{1,1} & c_{1,2} & \dots & c_{1,n} \\ c_{2,1} & c_{2,2} & \dots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & \dots & c_{n,n} \end{bmatrix} \begin{bmatrix} \mathscr{E}(v_1) \\ \mathscr{E}(v_2) \\ \vdots \\ \mathscr{E}(v_n) \end{bmatrix} = \begin{bmatrix} \Delta\mathscr{S}(v_1) \\ \Delta\mathscr{S}(v_2) \\ \vdots \\ \Delta\mathscr{S}(v_n) \end{bmatrix}$$

This can be rewritten as $C\mathscr{E} = \Delta\mathscr{S}$ which implies $\mathscr{E} = C^{-1}\Delta\mathscr{S}$. Thus, given a saliency field $\Delta\mathscr{S}$, the enhancement operator C^{-1} will generate the emphasis field $\mathscr{E}$. There are two parameters that govern the stability of the inversion of the matrix C. The first parameter is the relation between w_1 and w_2. Using the same values ($w_1 = w_2$) makes the matrix C rank-deficient because the sum of each row is zero. To alleviate the rank-deficiency we use unequal weights: $w_1 = 3/4$ and $w_2 = 1/4$. The second parameter is the scale σ and we discuss its effect on the matrix C in Section 4.3.2.

4.3.2 Emphasis Field

We have defined the enhancement operator C^{-1} which can generate an appropriate emphasis field for a given saliency field at a scale σ. Just as the saliency computation is based on the center-surround mechanisms at multiple scales, we would like to use enhancement operators at multiple scales σ_i. Let $\mathscr{E}_i$ be the emphasis field at scale σ_i. We compute this by applying the enhancement operator C_i^{-1} on the saliency field $\Delta\mathscr{S}$. Then, the final emphasis field may be computed as the summation of $\mathscr{E}_i$. These steps are illustrated in Figure 4.3.

For simplicity, we discuss the 1D binary case here. Consider the saliency field that is 1 over a desired emphasis region of length $2r$ and 0 everywhere else. This is shown in

Figure 4.3(b). We start by applying the enhancement operator C_1^{-1} at scale $\sigma_1 = (\sqrt{2}/8)r$ on the saliency field $\Delta\mathscr{S}$. We consider a geometric sequence of scales $\sigma_i = 2^{i-1} \cdot (\sqrt{2}/8)r$ while $\sigma_i \leq \sqrt{2}r$.

We have observed that the matrix C is well-conditioned for small values of σ that result in a diagonally dominant form. As the value of σ is increased, the matrix C ceases to remain diagonally dominant and in fact becomes close to singular. To address this, we sub-sample the saliency field by factors of $r/4$ ($\Delta\mathscr{S}_1$), $r/2$ ($\Delta\mathscr{S}_2$), r ($\Delta\mathscr{S}_3$), etc. and construct the appropriate matrices C_i. We then compute $\mathscr{E}_i = C_i^{-1}\Delta\mathscr{S}_i$ (shown in Figure 4.3(c)) and sum $\mathscr{E}_i$ to get the overall emphasis field $\mathscr{E} = \sum_{i=1}^{k} \mathscr{E}_i$ as shown in Figure 4.3(d). Note that we super-sample the sub-sampled fields so that the summation of all emphasis fields is carried out at the original scale.

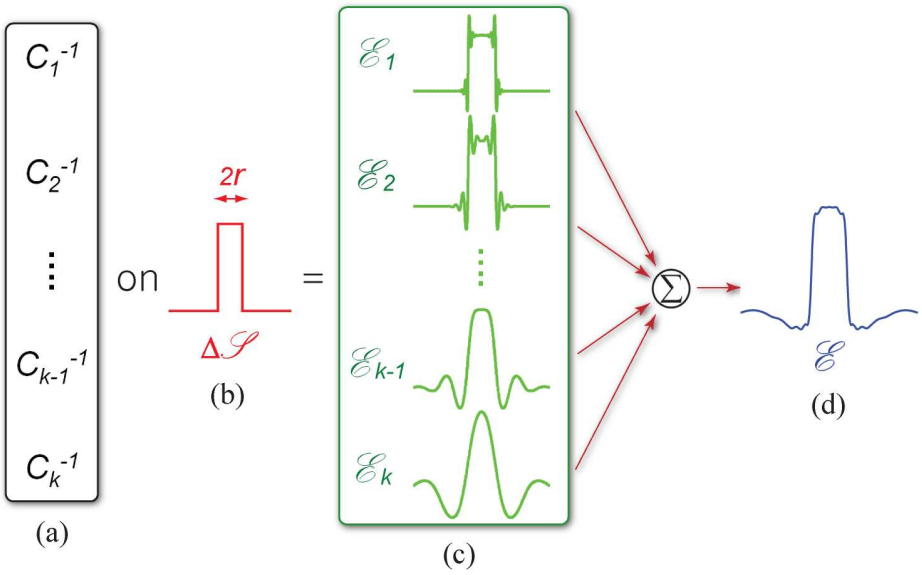


Figure 4.3: *Enhancement operator at scale σ_i is denoted by C_i^{-1} in (a). Figure (b) shows an example of saliency field with a desired emphasis region of length $2r$. The application of enhancement operator C_i^{-1} on saliency field $\Delta\mathscr{S}$ gives an emphasis field $\mathscr{E}_i$ in (c). Multi-scale summation of emphasis fields $\mathscr{E}_i$ generates the overall emphasis field $\mathscr{E}$ in (d).*

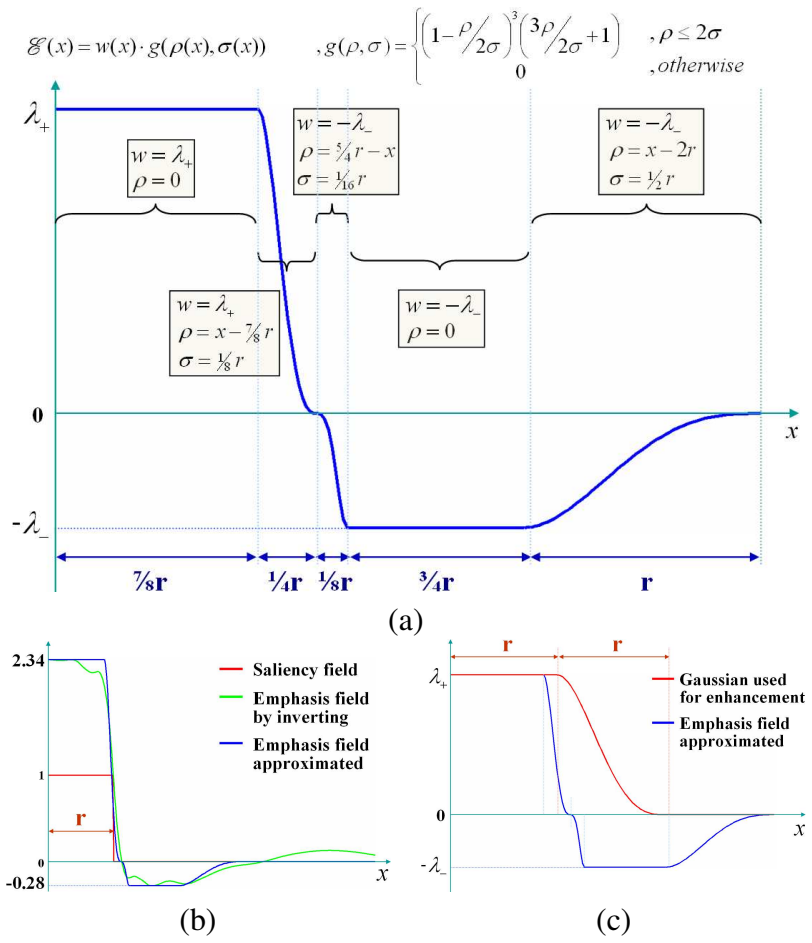


Figure 4.4: *Saliency-guided emphasis field. Here r denotes the radius of the user-specified region. Figure (a) shows the emphasis field approximated by a piecewise polynomial function. Figure (b) shows the emphasis field generated by multi-scale summation of emphasis fields in green and the approximation of it in blue with* $\lambda_+ = 2.34$ *and* $\lambda_- = 0.28$ *for comparison. Figure (c) shows the Gaussian that is used for enhancement for results in Figure 4.10.*

4.3.3 Emphasis Field in Practice

A system of simultaneous linear equations in n variables may be solved in time varying from $O(kn^2)$ for the Gauss-Seidel method where k is number of iterations to $O(n^3)$ for the Gaussian elimination method. For our experiments we were interested in enhancing saliency over regions that ranged in size from $n = 128 \times 256 \times 256$ to $n = 352 \times 352 \times 256$. This clearly would have been computationally very expensive. To address

this, we solve a 1D system of equations over $n = 640$ and assuming a spherical region of interest (ROI), interpret the results to be along the radial dimension. The 1D solution is shown in Figure 4.3(d) and by the green curve in Figure 4.4(b). Radial functions have been expressed using Gaussians, quadratic and higher-degree polynomials [9]. Here we use piecewise polynomial radial functions inspired by Wendland [129] to approximate the results. Our approximating function is shown in Figure 4.4(a) and by the blue curve in Figure 4.4(b). Figure 4.4(c) shows the Gaussian fall-off function from the boundary of the specified region with $\sigma = r/2$. The enhancements generated by this Gaussian are used for the comparisons in Section 4.4 and 4.5.

4.4 Visualization Enhancement

Once we have computed the emphasis field we can use it to modulate the various visualization parameters. Here we first discuss changing just one of these parameters – the brightness of a voxel as determined by the Value parameter in the HSV color model.

Brightness has been regarded as one of the most important components of color for drawing visual attention. Artists like Rembrandt and Caravaggio have skillfully used luminance contrast to emphasize key regions and characters in a painting. With our saliency-guide enhancement field $\mathscr{E}(\mathbf{v})$ at a voxel $\mathbf{v}$, we can easily modulate its brightness value V as:

$$V_{\text{new}}(\mathbf{v}) = V(\mathbf{v}) \cdot (1 + \mathscr{E}(\mathbf{v}))$$

where $-\lambda_{-} \leq \mathscr{E}(\mathbf{v}) \leq \lambda_{+}$. In the current implementation, we have used $0.4 \leq \lambda_{+} \leq 0.6$ and $0.15 \leq \lambda_{-} \leq 0.35$. Figure 4.5 compares the enhancement by a traditional Gaus-

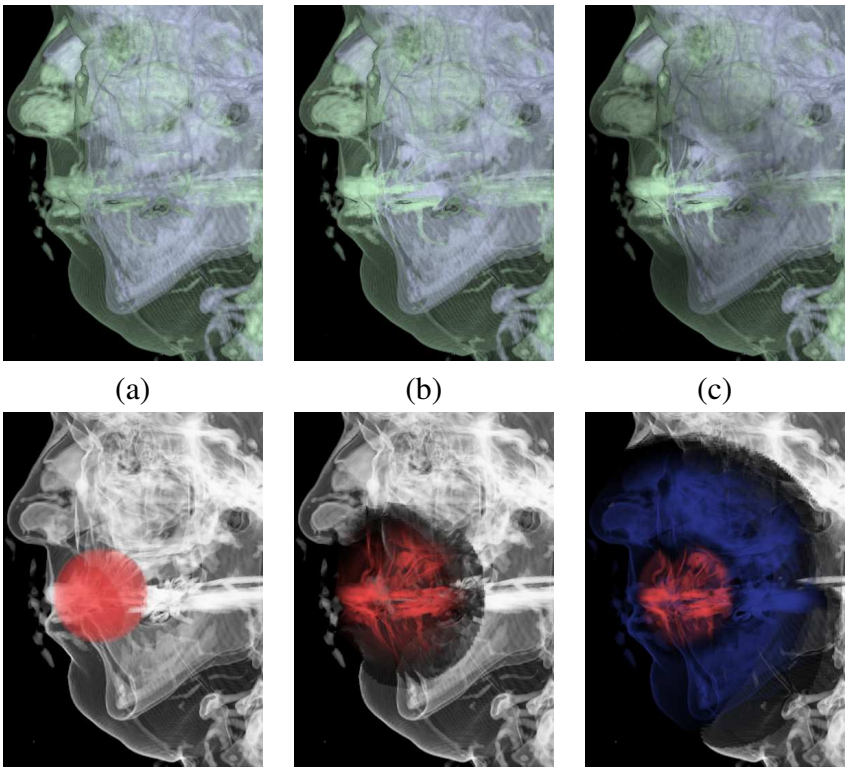


Figure 4.5: *The Visible Male model (*$128 \times 256 \times 256$*) is rendered by the traditional volume visualization in image (a). Images (b) and (c) show the visualization with regional enhancement by a Gaussian and our saliency-guided operator, respectively. User-specified spherical region of interest is shown in red in image (d) with a radius* $r = 20$*. Image (e) shows the Gaussian enhancement with* $\sigma = 10$*. Image (f) shows the emphasis field based on our method. The emphasis field value changes from* $\lambda_+ (= 0.4)$ *to 0 to* $-\lambda_- (= -0.15)$ *are represented by the color changes from red to black to blue. The radii of the spherical regions affected by the Gaussian-based and our method are* 40 *and* 60*, respectively.*

sian operator and by our new saliency-guided enhancement operator on the Visible Male model. Notice that the original image has high brightness regions such as the nose. While the Gaussian operator increases the brightness of the user-specified regions, our saliency-enhancement operator additionally lowers the brightness in the neighborhood. This difference results in a much greater user attention to the desired regions, even with subtle changes to the overall brightness. Figure 4.6 shows another comparison on the Engine Block model.

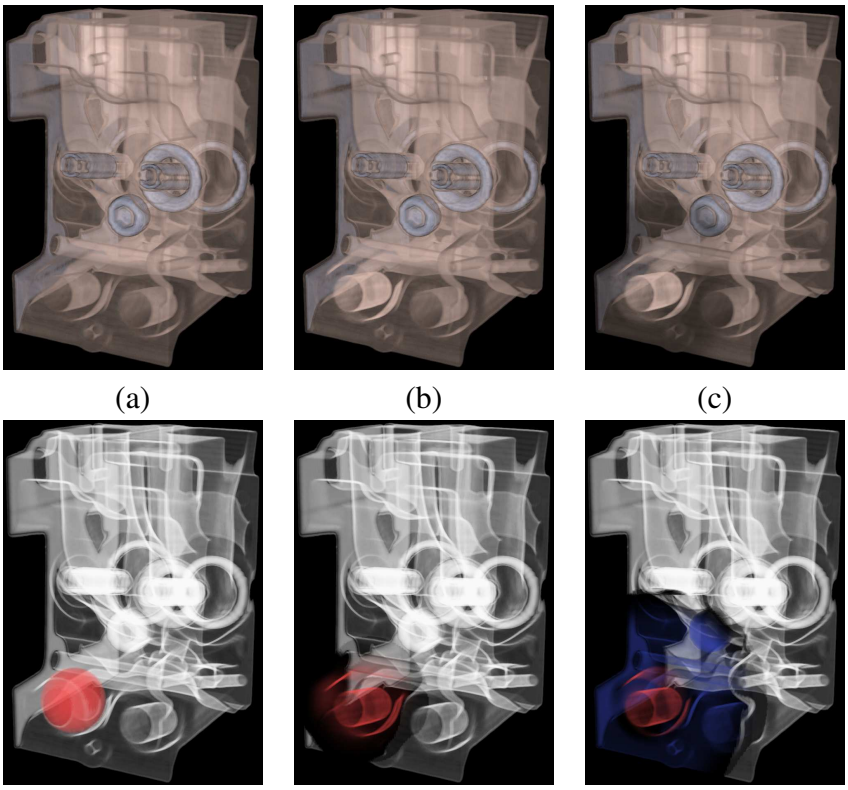


Figure 4.6: *The Engine Block model (*$256 \times 256 \times 256$*) is rendered by the traditional volume visualization in image (a). Images (b) and (c) show the visualization with regional enhancement by a Gaussian and our saliency-guided operator, respectively. User-specified spherical region of interest is shown in red in image (d) with a radius* $r = 20$*. Image (e) shows the Gaussian enhancement with* $\sigma = 10$*. Image (f) shows the emphasis field based on our method. The emphasis field values change from* $\lambda_+ (= 0.55)$ *to 0 to* $-\lambda_- (= -0.35)$*. The radii of the spherical regions affected by the Gaussian-based and our method are* 40 *and* 60*, respectively.*

Visual saliency can be increased by enhancing color saturation as well as the brightness. In cases where the brightness is already very high, it could be helpful to draw greater visual attention by enhancing color saturation. For instance in Figure 4.7(a) increasing the brightness any further will diminish the appearance of blood vessels at the center of the Sheep Heart. However a simple change in saturation can serve to draw visual attention as shown in Figure 4.7(b). Our technique can increase the overall color saturation in a way similar to what we have outlined above for brightness.

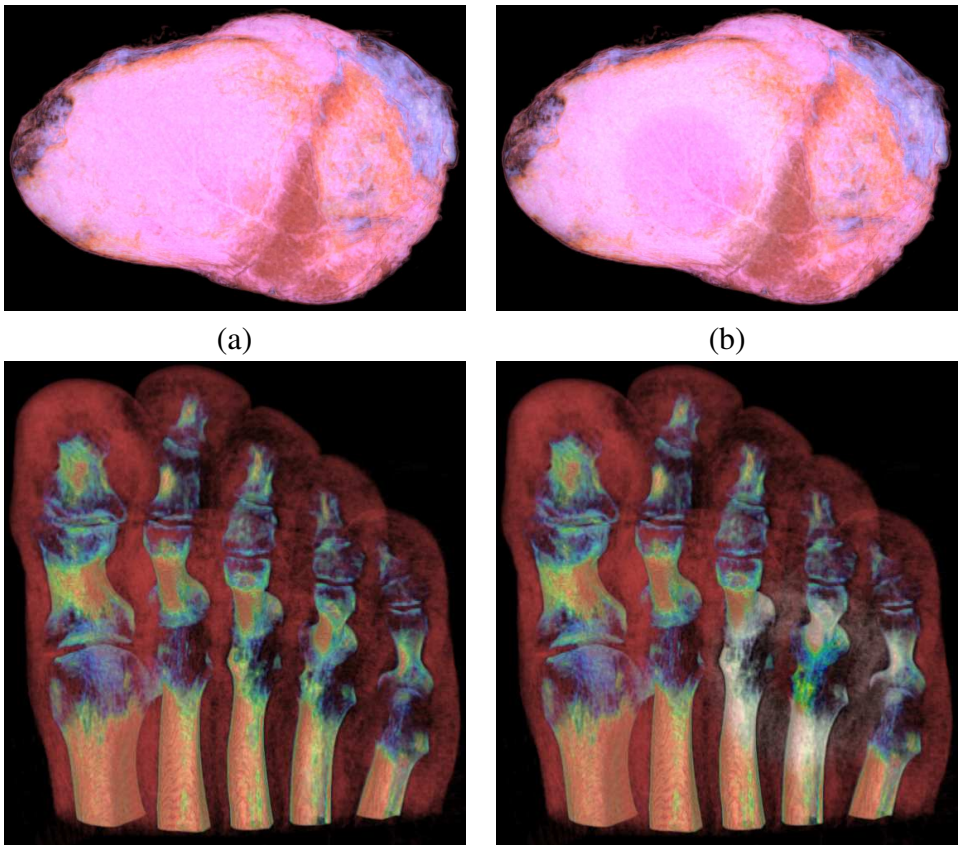


Figure 4.7: *Saturation enhancement for the Sheep Heart and the Foot models. Images (a) and (c) show the tranditional volume visualization. Images (b) and (d) show the visualization with color saturation enhancement based on our saliency-guided enhancement operator applied on the blood vessels at the center and the fourth toe, respectively in each model.*

4.5 User Study

We have carried out an eye-tracking-based user study to gather objective evidence of the effectiveness of our approach. Our goal in this user study is to validate our ability to draw a viewer's attention by subtle changes to the appearance of the volume data.

4.5.1 Hypothesis

Our first hypothesis is that the eye fixations increase over the region of interest in a volume by the saliency-guided enhancement compared to the traditional volume visualization. Our second hypothesis is that the eye fixations increase over the region of interest in a volume by the saliency-guided enhancement compared to the Gaussian-based enhancement. We would like to validate both of the above hypotheses with a visually subtle level of enhancement.

4.5.2 Experimental Design

We carried out the study with 10 subjects that had normal or corrected-to-normal vision and who were not familiar with this work. General settings of eye-tracker and its calibration are similar to those explained in Section 2.3.

Duration: The user study had 12 trials (images). Each trial started with the subject seeing a blank screen with a cross at the center of the screen. The subject was asked to look at the cross before clicking the mouse to bring up the next image. This ensured that each trial started with the subject's eyes fixated at the center of the image. Each image was shown for 5 seconds. Each study took about 80 seconds. Subjects were told to freely view the images with no assigned goal and were informed in advance about the design of each trial including the duration each image would be shown and the total number of images.

Image Ordering: There were a total of 20 images used in all the experiments. Each image set consists of one original image and four enhanced images in which one

of two regions is enhanced by either a Gaussian or our saliency enhancement. We have used the volume datasets of the Engine Block, the Foot, the Visible Male, and the Sheep Heart model for our study. Each user saw 12 images out of these 20 images. When we ordered the images for each user, we considered differential carryover effects and the counter-balancing problem for the pairwise analysis on the results. First, we placed similar images far apart to alleviate differential carryover. At the same time, we did not place the similar images in perfectly regular manner so that a user could not guess what image will be shown next. Alleviating differential carryover effect had the highest priority in our ordering because a user is supposed to look at 3 similar images (original, enhancement on two different regions) for each model. Second, each user looked at two images where we enhanced different regions with different types of operators (Gaussian-based and Saliency-guided). Finally, we randomized the order of regions and the order of enhancement types (Gaussian and saliency-based) to counterbalance overall effects.

4.5.3 Data Analysis

The results of our study are shown in Figure 4.8. Each grouping of bars shows the percentage of fixations that fell in a desired region for the unaltered, Gaussian-enhanced, and Saliency-enhanced visualizations for a specific model and region on that model. Figure 4.9 shows the increase in the number of fixation points on the region of interest after the application of the saliency-guided operator.

First, we analyzed the effects of each enhancement technique on two different regions for each model. We have analyzed the differences in fixations between the first

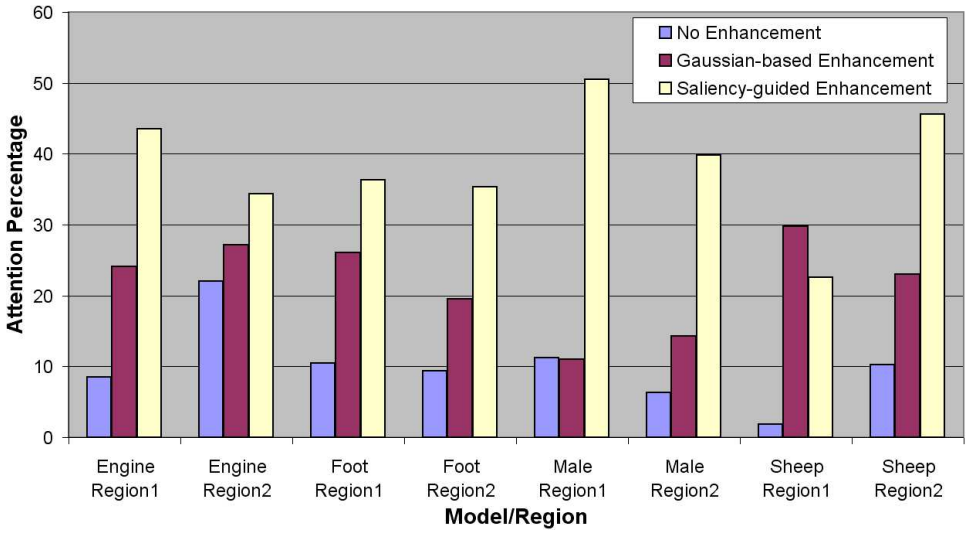


Figure 4.8: *Fixation results for volume visualization enhancements.*

region and the second region of each model for the three cases – (a) unaltered, original visualization, (b) first region is enhanced, and (c) second region is enhanced. We carried out pairwise t-tests with the assumption that each region of interest occupies the same number of image pixels and has a similar percentage of fixations in the original model. This assumption did not hold for the Engine Block model since one region turned out to be brighter than the other. Also, it did not hold for the Sheep Heart model since one region was closer to the center and drew greater fixations [91]. Therefore, we did not include these two in Table 4.1. As the results show, we did not observe significant differences in the percentage of fixations when a region was enhanced by the Gaussian-based method in any of cases. However, we can clearly observe significant differences in all cases when a region is enhanced by the Saliency-guided method.

We next carried out a pairwise t-test on the percentage of fixations before and after we apply enhancement techniques for each model (this is the only condition in the test). Table 4.2 shows the results from all the models. We found a significant difference in the

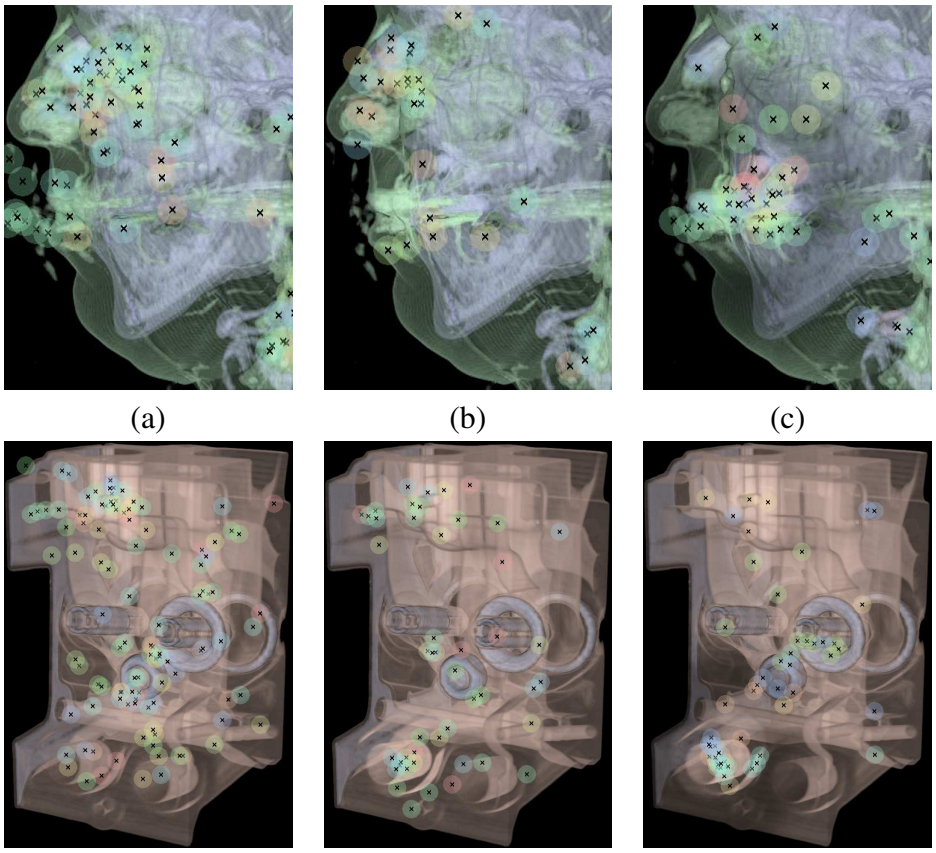


Figure 4.9: *Images (a), (b), (c) and images (d), (e), (f) show the fixation points collected from our user study with the images generated from the Visible Male model and the Engine Block model, respectively. The images (a) and (d) were rendered using the traditional volume visualization. The images (b) and (e) were enhanced by a Gaussian operator while the images (c) and (f) were enhanced by our saliency-guided operator, respectively.*

percentage of fixations when we applied saliency-guided enhancement for all the models. There was a difference for the percentage of fixations when we applied Gaussian-based enhancement for all the models other than the Visible Male model.

When the results from each region alone in Visible Male and Sheep Heart were analyzed by pairwise *t*-tests with saliency-guided technique condition, there also was a significant effect, (t-value=-7.35, p=0.002 for Visible Male). However for the Engine Block and the Foot model, there was only a borderline significant effect by saliency-

Table 4.1: *Pairwise t-tests on the 1st and the 2nd Areas of Interest.*

Model	Condition	t-Value	p-Value
Foot	No Change	0.312	0.762
	Region1 enhanced by Gaussian	1.35	0.248
	Region1 enhanced by Saliency	2.74	0.052
	Region2 enhanced by Gaussian	-0.68	0.534
	Region2 enhanced by Saliency	-2.96	0.042
Visible Male	No Change	0.959	0.363
	Region1 enhanced by Gaussian	1.34	0.25
	Region1 enhanced by Saliency	4.39	0.012
	Region2 enhanced by Gaussian	-0.57	0.601
	Region2 enhanced by Saliency	-5.82	0.004

Table 4.2: *List of pairwise t-tests.*

Model	Condition: No Change vs.	t-Value	p-Value
Engine Block	Gaussian-based enhancement	-2.36	0.042
	Saliency-guided enhancement	-2.86	0.019
Foot	Gaussian-based enhancement	-2.67	0.026
	Saliency-guided enhancement	-3.34	0.009
Visible Male	Gaussian-based enhancement	-0.661	0.525
	Saliency-guided enhancement	-6.65	< 0.001
Sheep Heart	Gaussian-based enhancement	-3.86	0.005
	Saliency-guided enhancement	-4.49	0.002

guided technique. We can only observe a significant effect when the results from each region alone in Sheep Heart were analyzed with Gaussian-based technique. We think that this is due to the small number of observations. We believe those results would also be significant if there were more participants because there was a clear trend showing an improvement on all models in Figure 4.8.

4.6 Conclusions and Future Work

We have developed a saliency-based enhancement of volume visualization and successfully validated its ability to elicit viewer attention. Our model is inspired by the center-surround mechanisms of the human visual system. We have found that it is more successful at eliciting viewer attention than the traditional Gaussian regional enhancement approach. Saliency-guided enhancement for volume visualization can be helpful in several contexts. For instance, our approach could be used in helping users navigate through complex volumetric datasets and facilitating their understanding by guiding their attention to regions and objects selected by a domain expert. It will also be interesting to examine the applicability of this approach for exploratory visualization systems that rely on automated and fuzzy detection of features. Such systems could use saliency-based perceptual enhancement to generate a variable level of perceptual interest to the human observer. At present we have explored saliency-guided alteration of brightness and color saturation for volumes. In future we plan to also explore the implications of this framework for other appearance attributes such as opacity and texture detail. Our approach at this time has been validated only on static volumetric scalar field datasets. It will be interesting to generalize it further to be able to handle time-varying datasets with multiple superposed scalar and vector fields.

Our current method has been validated on spherical regions of interest and binary-valued saliency field. Generating an emphasis field from an arbitrary-shaped region with general saliency values will be considered in the future. At this time we do not have any evidence if our approach can actually enhance the comprehensibility of the volume

rendered images. We will like to further study this in the future. Visual saliency is very sensitive to scale. Identifying the appropriate scales and their relative importance is another valuable area for future research in guiding visual attention.

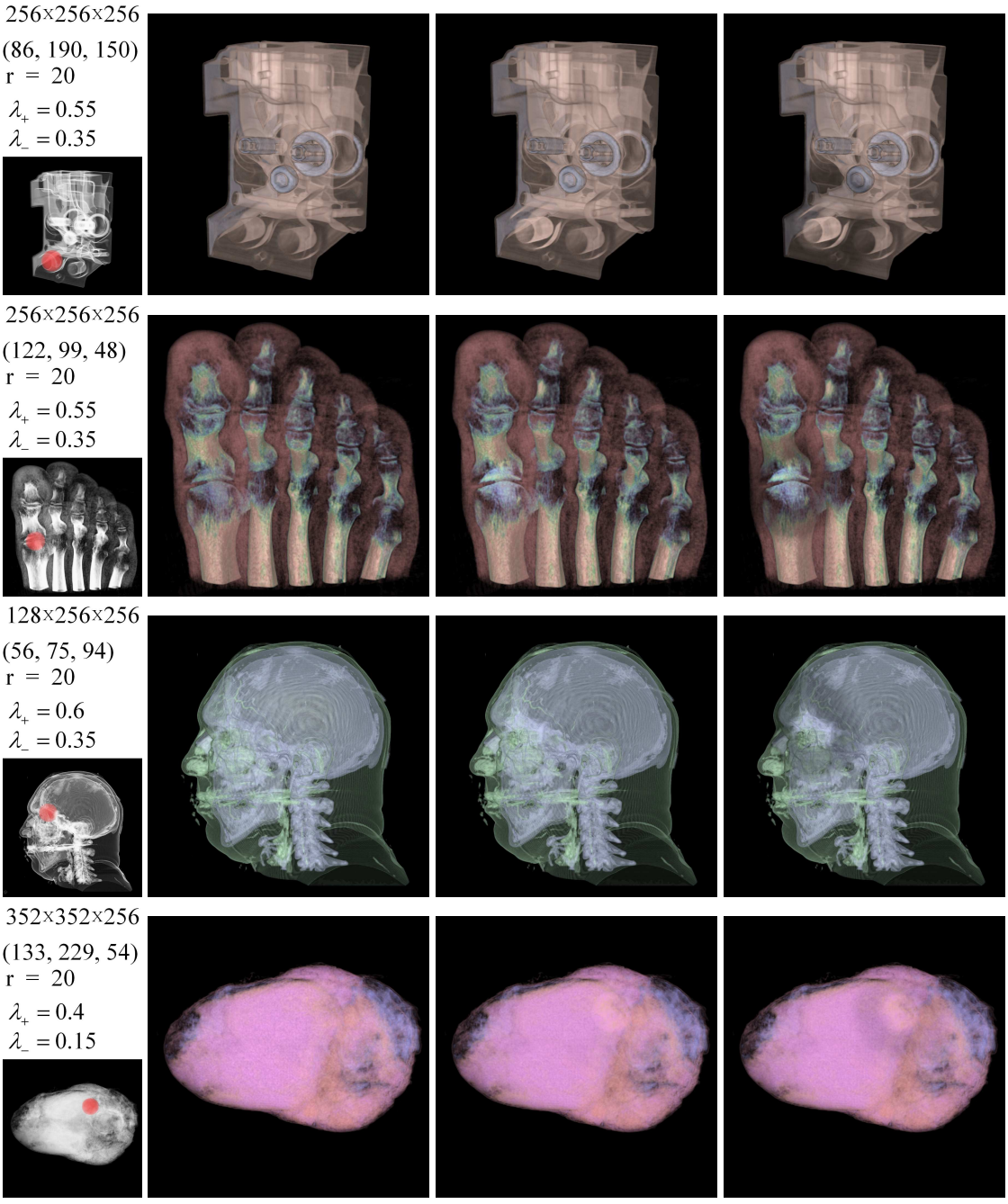


Figure 4.10: *A subset of images used in the user study. The first column shows the saliency fields, the spherical regions of interest (ROI) marked in red. We also show the size of each volume dataset, the center of ROI, the radius of ROI, and each of weights,* λ_+ *and* λ_- *used for enhancements above each saliency field image. The second column shows the traditional volume rendering. The third column shows the visualization with value enhancement in HSV color model based on the Gaussian-based enhancement while the fourth column shows the visualization with value enhancement based on our saliency-guided enhancement.*

Chapter 5

Salient Transformation Streams

In a number of real-life datasets, we observe that certain local geometries may appear in the same configuration in another part of the model. These self-similarities are obvious for architectural CAD (repeated doors, windows, furniture), mechanical CAD (repeated sub-assemblies, bolts, cylinders), molecular CAD (repeated amino acids, nucleic acids, alpha helices, beta sheets), and terrain layouts for games (trees, grass, flowers). We believe looking for non-repeating structure in the middle of repeating structure is another way to identify the salient regions in these datasets. In this chapter, we efficiently encode such repeating patterns to increase interactivity for 3D graphics and visualization.

Our salient transformation streams idea was also inspired by recent trends in parallel computer architecture. These trends strongly suggest the need to improve the arithmetic intensity (the compute-to-bandwidth ratio) for greater performance in time-critical applications, such as interactive 3D graphics. At the same time, advances in stream programming abstraction for graphics processors (GPUs) have enabled us to use parallel algorithm design methods for GPU programming.

In this chapter, we explore the interactions between multiple data streams to improve arithmetic intensity and address the input geometry bandwidth bottleneck for interactive 3D graphics applications. We introduce the idea of creating vertex and transformation streams that represent large point data sets via their interaction. We discuss

how to factor such point datasets into a set of source vertices and transformation streams by identifying the most common translations amongst vertices. We accomplish this by identifying peaks in the cross-power spectrum of the dataset in the Fourier domain. We validate our approach by integrating it with a view-dependent point rendering system and show significant improvements in input geometry bandwidth requirements as well as rendering frame rates.

5.1 Related Work

The graphics community has long faced the challenge of interactively exploring very large 3D graphics models while reconciling the mutually conflicting goals of scene realism and interactivity. A crucial bottleneck in this has been the input geometry bandwidth. Consequently, there has been a long history of work related to reduction of the geometry bandwidth to the graphics processor to achieve greater interactivity.

Triangle strips and triangle fans are amongst the earliest data-structures designed to address the input geometry bandwidth bottleneck. Although each triangle can be specified by three vertices, to maximize the use of the available data bandwidth it is desirable to order the triangles so that consecutive triangles share an edge. Such ordered sequences of triangles are referred to as triangle strips or triangle fans. Using such an ordering, only the incremental change of one vertex per triangle needs to be specified. These methods require sending $n+2$ vertices for n triangles, instead of $3n$ vertices, potentially reducing the input geometry requirements by a factor of three.

Visibility-based culling and level-of-detail-based rendering reduce the input geom-

etry by reducing the number of graphics primitives. Visibility-based culling schemes only send those primitives to the graphics processor that are visible or potentially visible [17]. Level-of-detail-based rendering schemes send simpler, lower fidelity representations of an object whenever higher complexity representations are deemed unnecessary – such as when the object is being displayed on a small number of pixels on the screen or is otherwise perceptually less important [77].

Recent improvements in scene acquisition techniques such as laser scanning and computer-vision-based sensing have resulted in a growing collection of 3D graphics datasets that are based on points. Consequently, point-based rendering schemes [38, 71, 73, 96, 107, 109] have evolved as a viable alternative to triangle-based representations. The point primitives can be rendered as spheres [107], points with attributes (Surfels) [96], tangential disks (Surface splats) [10, 41, 89, 102, 139], tangential ellipses [132], quadratic surfaces [57], higher degree (3 or 4) polynomials [4, 32], using wavelet basis [128], and octree cells [11, 96, 131]. These and several other techniques involve transmission of points and their attributes from the CPU to the GPU every frame. Points can also be rendered without any CPU involvement by storing the point geometry directly on the graphics card [10, 20, 41]. Temporal coherence can be exploited by keeping track of the visible Surfels in the frame buffer of successive frames [40].

In some ways, the flavor of the approach presented in this dissertation is closer to that of triangle strips and triangle fans, that involve re-ordering the input list of primitives to succinctly represent them as a single data stream. However, it differs from all previous work in that it addresses the geometry bandwidth bottleneck by harnessing the power of multiple interacting streams of data, instead of a single stream. Our approach is comple-

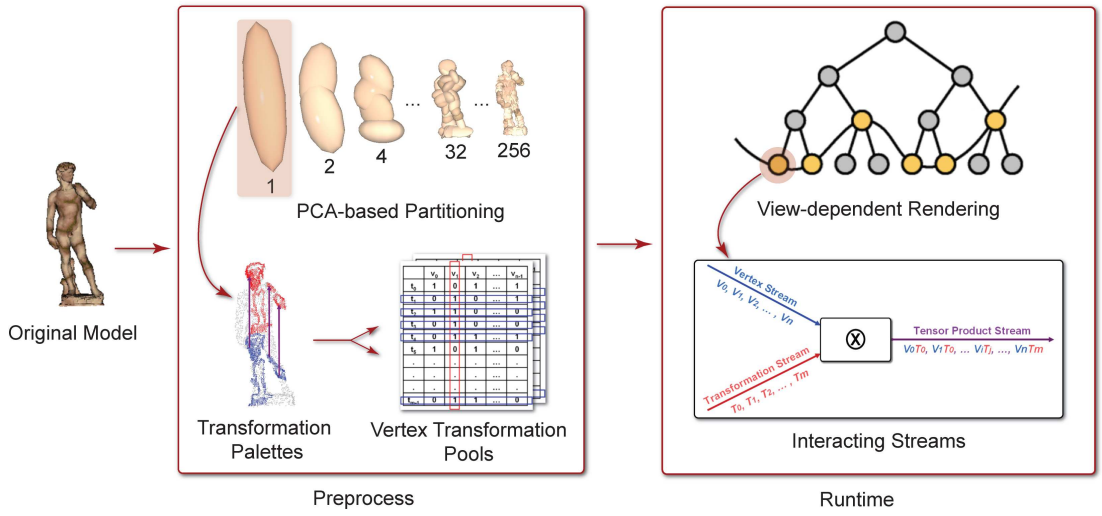


Figure 5.1: *Overview of Salient Transformation Streams. As a preprocess, the PCA analysis of a group of points results in a binary partition tree. Each node of the tree is divided into its child nodes depending on the positions at its local orientation frame. In transformation palettes stage, we identify the most common translations among the vertices in each node. Then, we construct two vertex transformation pools by which we cover as many vertices as possible. At runtime, a view-dependent manager determines the cut in the tree and each node of the cut is visualized by the interaction between vertex and transformation streams.*

mentary to, and can augment, existing schemes such as level-of-detail-based rendering that reduce the number of geometry primitives to be rendered.

5.2 Overview

Figure 5.1 shows the overview of our salient transformation streams. We introduce the idea of interacting vertex and transformation streams to encode general point cloud datasets and discuss how these streams can be decoded using modern vertex shaders in Section 5.3. We discuss how to efficiently build vertex and transformation streams from a pool of paired vertices and transformations in Section 5.4. In Section 5.5, we outline a method to identify the most common transformations that can map a set of vertices to

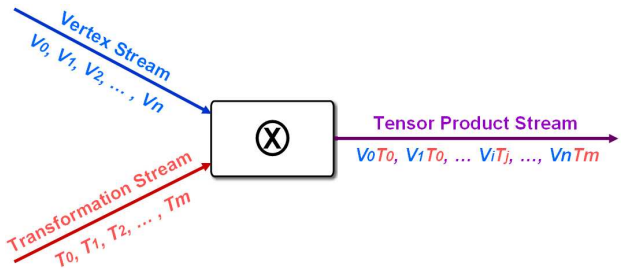


Figure 5.2: *Stream Interactions: 'Vertex Stream' contains the mesh vertices, 'Transformation Stream' contains instance transformations that will act on the vertices in the vertex stream. The two streams are combined on the GPU and generate the 'Tensor Product Stream' which has the output vertices for rendering.*

itself using the Fast Fourier Transform. Finally in Section 5.6 and 5.7, we show how our approach of using transformation streams can improve the arithmetic intensity in a view-dependent rendering application.

5.3 Interacting Streams

Our goal in factoring an input list of vertices into two interacting streams of vertices and transformations is to reduce the input geometry bandwidth requirements and improve the arithmetic intensity of the participating streams. As shown in Figure 5.2, these two streams – the vertex stream and the transformation stream – can then be combined with each other on a GPU resulting in an output stream of vertices that is a tensor product of the input streams. Thus, in the best case, we might be able to factor n vertices into two streams of size $\sqrt{n}$ each, thereby reducing the input bandwidth requirements by a factor of $O(\sqrt{n})$.

Consider an idealized geometry of 16 points shown by spheres in Figure 5.3. Let the four white points comprise the vertex stream. Then using a set of four translations as shown in the figure, one can generate all the input points. We include the null translation

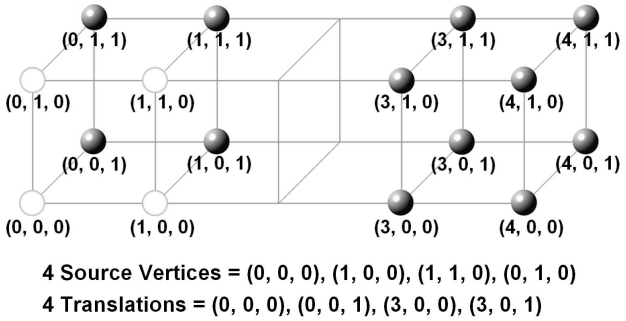


Figure 5.3: *The vertex stream has the four white source vertices and the transformation stream has four translations. Each of the twelve black vertices can be reached by applying one of the three non-null translations to the white vertices.*

$(0,0,0)$ for completeness.

The vertex transformation streams discussed above are ideal. In practice it is rare to find such a perfect mapping between vertices and transformations. Even when we did find such mappings, they covered very small sets of vertices. To get larger interaction streams we decided to generalize our interactions between vertices and transformations. Instead of insisting that every vertex interacts with every transformation, we allowed some vertex transformation pairs to not contribute any vertex to the output stream. This simple generalization allowed us to greatly enhance the scope and size of the vertex transformation streams that our approach could identify. In Figure 5.4 we show the relationships between translations and vertices of two sets of geometries. A 1 indicates that the translation in the row interacts with the vertex of the column while a 0 indicates non-interaction. Our generalization allows us to construct interacting streams that can represent the slightly irregular geometry of Figure 5.4(b).

We have implemented stream interaction on current-generation GPUs using the geometry instancing features of the latest vertex shader model (`VS 3.0`). Consider the case of *n* vertices in stream 0 and *m* transformations in stream 1 as shown in Figure 5.2. To use

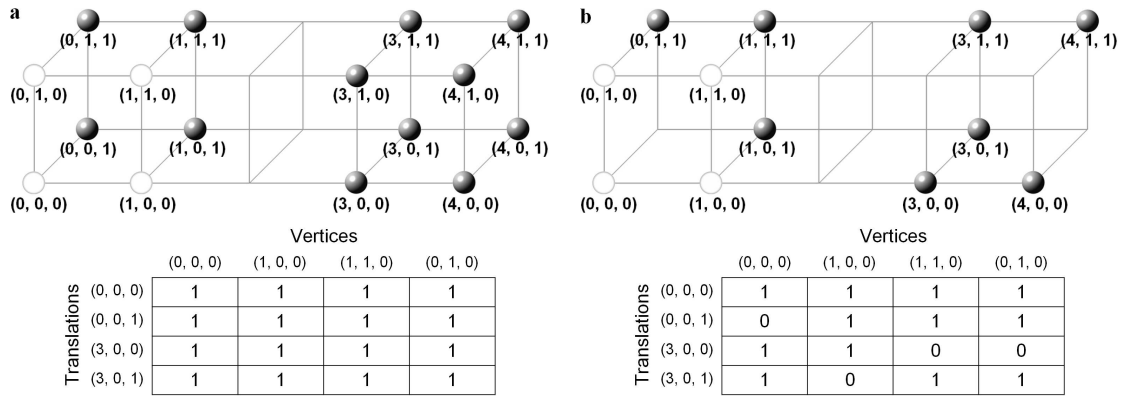


Figure 5.4: *Interactions between the vertex stream and the transformation stream are represented by binary tables. In these vertex-transformation tables, a* 1 *indicates interaction between a vertex and a translation and* 0 *indicates no interaction. Figure (a) shows the table for the idealized point set of Figure 5.3. Figure (b) shows another point set and its vertex-transformation table.*

the geometry instancing feature of (`VS 3.0`) we set the frequency of the vertex stream to m and the frequency of the transformation stream to 1. As shown in Figure 5.2, the vertex shader is first invoked with V_0T_0. This is followed by V_1T_0, V_2T_0, and so on. After all the vertices for the first transformation(T_0) have been processed, the pointer to vertex stream (stream 0) is reset and the pointer to transformation stream (stream 1) is incremented to T_1 [28].

In our implementation, we specify the frequency of the vertex stream to be the number of transformations in the transformation stream. Since the geometry instancing only allows us to achieve all-pairs interactions between the elements of the two streams, we encode the interaction information between the transformation and vertex streams using tags that we pass with the elements of each stream. The transformation stream's tag is just a short integer which indicates the index of that transformation in its stream. Thus the i^{th} transformation's tag has value i. A vertex's tag is an occupancy bit vector

```
VS_OUTPUT InstancedVS (
    short3 vPos : POSITION,
    int vertexTag : TEXCOORD0,
    short3 vTrans : TEXCOORD2,
    int transTag : TEXCOORD3 )
{
  VS_OUTPUT Output;
  Output.Color = float4(1,0,0,1);

  int mod = vertexTag % transTag
  if (mod >= transTag / 2)
  {
    // Object Coord
    Po = float4(vPos.xyz + vTrans.xyz, 1.0);
    // Screen Coordinate
    Output.Position = mul(Po, mWorldViewProj);
  }
  else
  {
    Output.Position = float4(1.0, 1.0, 1.0, 0.0);
  }
  return Output;
}
```

Figure 5.5: *Pseudo-code for a Vertex Shader program shows how we use vertex and transformation tags to determine if a pair of elements across vertex and transformation streams should interact, and if so how to generate the output stream vertices.*

that encodes the translations that the vertex should interact with. Therefore, if we use an unsigned integer as the tag for a vertex, we can encode its interactions with up to 32 translations. Figure 5.5 shows the pseudo code of the vertex shader program for checking and implementing such interactions. Note that if the interaction between a vertex and a transformation is not supposed to happen, our code sets the output vertex to infinity (in homogenous coordinates) and the vertex is then culled away. Unfortunately, the current generation of GPUs do not support bit-wise integer operations. As a result we had to emulate these bit operations by using floating-point operations that were available to us.

5.4 Vertex Transformation Pools

In the previous section we discussed how we can generate an output stream of vertices as a tensor product of vertex and transformation streams. We also discussed how we can fine-tune (allow or disallow) interactions between the two streams by appropriately tagging the elements of the two streams and using vertex shader programs. In this section we shall discuss how to identify such streams from raw input point datasets.

Consider a model with n points, $\vec{p}_i$, $0 \leq i < n$. For any pair of points $\vec{p}_i$ and $\vec{p}_j$, $0 \leq i, j < n$, consider a transformation that maps $\vec{p}_i$ to $\vec{p}_j$. In general, there are an infinite number of such transformations. Therefore, let us restrict ourselves to the set of translations. Let translation $\vec{t}_{ij}$ be specified as $\vec{t}_{ij} = \vec{p}_j - \vec{p}_i$. For n points, we can identify n^2 such transformations. Now, for most real-life datasets we have observed that out of these n^2 transformations only $m << n^2$ are unique. We discuss some reasons for this transformation space coherence and give a method to identify such unique transformations in Section 5.5. Now consider a large *vertex transformation table* whose columns are n vertices and whose rows are the m unique transformations. The $(i, j)^{th}$ element of this table is a boolean value which is set to 1 iff $\vec{t}_i + \vec{p}_j = \vec{p}_k$ for some $k < n$. That is, if the i^{th} translation maps the j^{th} vertex to some other vertex in the input data, we flag this as an interaction we permit. Otherwise we set the $(i, j)^{th}$ entry's boolean value to 0. If $\vec{t}_i + \vec{p}_j = \vec{p}_k$, we say that point $\vec{p}_j$ *covers* point $\vec{p}_k$.

In order to get large vertex and transformation streams, our goal is to find the largest rectangle in the vertex-transformation matrix (after reordering rows and columns) such that the fraction of ones in it is higher than some threshold (δ). We refer to such a maxi-

mal reordered rectangle as a *vertex transformation pool*. Large values of δ tend to return very small vertex-transformation pools whereas small values of δ result in too many non-interacting vertex-transformation pairs which will later require culling. We have observed that $\delta = 0.5$ reconciles these goals well. Identification of the vertex transformation pools is an iterative process. After we find a vertex transformation pool, we zero-out its entries in the vertex transformation table, and repeat the process to get the next-best vertex transformation pool.

5.4.1 Finding the First Vertex Transformation Pool

At first glance, the problem of finding good vertex transformation pools resembles the edge-maximizing bipartite clique problem [94, 60], where the rows are one side of the graph, the columns are the other, and there is an edge between i and j if the $(i, j)^{th}$ entry's boolean value is 1. However, our problem is different because we would also like to include some 0 entries in the pool as long as it allows us to increase the overall fraction of 1's in the pool. A polynomial-time optimal solution to this problem seems unlikely, so we have used a greedy heuristic.

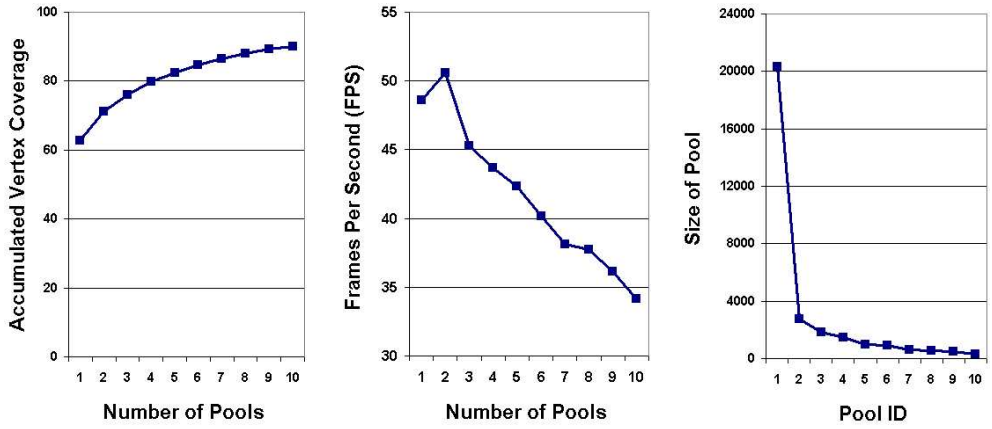
We first find the vertex $\vec{p}_j$ that has the most 1-values in its column. Let the number of transformations for $\vec{p}_j$ be k. We next restrict ourselves to the k rows for which the column for $\vec{p}_j$ has a 1. We then sort all the vertices by the sum of 1-values they have in these k rows and process them in the decreasing order of their sums. For each vertex $\vec{p}_i$, we determine the number of vertices it can cover that were previously not covered. If the number of newly covered vertices is greater than a threshold, we include $\vec{p}_i$ in the pool. In

all our experiments we have set the threshold to be 25% of the value of k for the current pool. We have observed that this gives us vertex-transformation pools with the fraction of ones in the pool, $\delta \approx 0.5$.

It turns out that there can be redundant coverage amongst the vertices in a pool. Thus, it is possible that $\vec{t}_i + \vec{p}_j = \vec{t}_{i'} + \vec{p}_{j'} = \vec{p}_k$. This is wasteful in that we might end up processing the same vertex multiple times. We handle this by not counting such previously covered vertices in our quest to maximize the size of our vertex transformation pools.

5.4.2 Updating for Subsequent Pools

After identifying a vertex transformation pool we update the vertex transformation table to discard the covered vertices. Since we prioritize the vertices based on the number of their 1-entries, the number of vertices that a given vertex covers is actually its importance level for inclusion in future pools. For each vertex $\vec{p}_k$ that our identified pool covers, we decrease the weights of all other vertices that could cover $\vec{p}_k$. Thus, for each covered vertex $\vec{p}_k$ we set all those (i, j) entries in the vertex transformation table to 0, for which $\vec{t}_i + \vec{p}_j = \vec{p}_k$. We can do this updating efficiently if each vertex maintains a list of all other vertices that it can cover. Thus, let $\vec{p}_k$ have the list of vertices $(\vec{p}_{k1}, \vec{p}_{k2}, \ldots, \vec{p}_{kl})$ that can be reached from itself using its translations $(\vec{t}_{k1}, \vec{t}_{k2}, \ldots, \vec{t}_{kl})$. Each of these covered vertices $\vec{p}_{ki}$ must have the reverse translation $(-\vec{t}_{ki})$ in its own translation lists. Therefore, we can decrease the importance of those covered vertices by one by just resetting the appropriate elements in the vertex transformation table.



(a) (b) (c)

Figure 5.6: *Tradeoffs between the number of pools, vertex coverage, and rendering time for Stanford's David model. Figure (a) shows the vertex coverage and figure (b) shows the rendering time as we increase the number of pools. Figure (c) shows the changes of the size of each vertex transformation pool, which is the product of the number of translations and the number of vertices in each pool.*

5.4.3 Determining the Number of Pools

To show how the number of pools affects the overall performance of this algorithm, we have plotted the effect of the number of pools against the vertex coverage and the final rendering time for the Stanford's David in Figure 5.6. As we increase the number of vertex transformation pools, they can cover more vertices, but the overall rendering performance drops. There are two reasons for this. First, using additional blocks incurs the overhead of using `drawCall( )`. We have observed that using too many `drawCall( )`s makes the applications CPU bound. Second, as we identify more pools, the size of each pool, which is the product of the translations and the vertices in it, gets smaller. For small pools, the overhead of using instancing for drawing outweighs the benefit of transferring less data to the GPU. Based on these considerations, we decided to select only the first two pools for our results. The vertices that are not covered by these two pools are rendered without any geometry instancing using conventional rendering.

5.5 Transformation Palettes

We have thus far discussed how to identify maximally-sized vertex-transformation pools and how to convert them into interacting vertex and transformation streams. In this section we discuss how to identify the most common transformations amongst vertices. As we discussed earlier in Section 5.4, if we consider a model with n points, $\vec{p}_i$, $0 \leq i < n$ and restrict ourselves to translations, it is possible to get n^2 transformations amongst all pairs of points. In the worst case, each of these n^2 transformations can be unique and we will not be able to benefit from a transformation-based coding of the input data. Fortunately, real data does have plenty of such coherence due to several factors – input data symmetries, coherence in procedural or simulation data generation, coherence due to acquisition device characteristics, and even coherence due to quantization algorithms. In a number of real-life datasets, we observe that certain local geometries may appear in the same configuration in another part of the model. These symmetries are obvious for architectural CAD (repeated doors, windows, furniture), mechanical CAD (repeated subassemblies, bolts, cylinders), molecular CAD (repeated amino acids, nucleic acids, alpha helices, beta sheets), and terrain layouts for games (trees, grass, flowers). Our algorithm can easily detect such symmetries in the transformation space. What is more interesting, however, is that for several cases, such repeated patterns might not even be visually obvious. We have observed that a large fraction of 3D point geometry representing real-life datasets from 3D scanners can be efficiently expressed in this way. In Figure 5.7 we show one example of a frequent translation our algorithm discovered in the Stanford's David model.

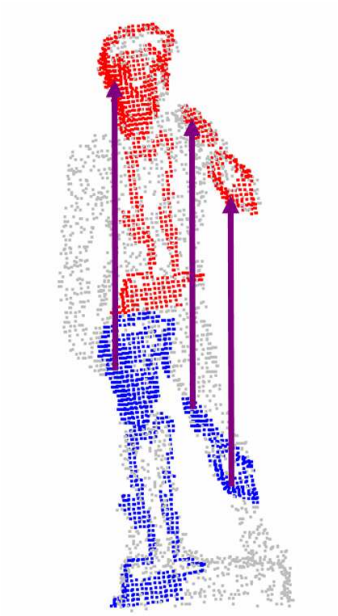


Figure 5.7: *An example of frequently occurring common translations of vertices identified by our algorithm in Stanford's David model. If we quantize the David model on a* 128^3 *grid to* $150K$ *quantized points we find the translation* $(62,-6,-5)$ *ocurring* 1261 *times as shown.*

The problem of finding the set of two point pairs which specify the same translation $\vec{t}$ can be reduced from 3SUM problem. The 3SUM problem can be solved by a simple $O(n^2)$ algorithm [35] and recent advances present a subquadratic randomized algorithm [7]. Using a finite precision model, a 3SUM hard problem can be solved in $O(n \log n)$ time using the FFT (Fast Fourier Transform) [19]. Here we used the FFT to find the common translations efficiently.

We explain our method in the simplified one-dimensional case. For a point set $P = \{p_0, p_1, p_2, ..., p_{N-1}\}$, $0 \leq p_i \leq M$, we can think of a polynomial $A(x)$ where the exponent of each term is the coordinate of each point, and we can think of another polynomial $B(x)$ where the exponent of each term is the negative value of the exponent of $A(x)$.

$$A(x) = \sum_{i=0}^{M} a_i x^i, \text{ where } a_i = \begin{cases} 1, \text{ if } i \in P \\ 0, \text{ otherwise} \end{cases} \tag{5.1}$$

$$B(x) = \sum_{i=0}^{M} a_i x^{-i}, \text{ where } a_i = \begin{cases} 1, \text{ if } i \in P \\ 0, \text{ otherwise} \end{cases} \tag{5.2}$$

Let $C(x)$ be the polynomial representing the multiplication of polynomials $A(x)$ and $B(x)$. The exponent of each term in $C(x)$ can be interpreted as the translation between two points in P, and the coefficient of that term indicates the frequency of occurrence of the translation. In other words, c_i is the number of points in P which can be translated from other points in P by translation i.

$$C(x) = \sum_{i=-M}^{M} c_i x^i \tag{5.3}$$

The multiplication of two polynomials can be computed in $O(n \lg n)$ time by converting polynomials into *point-value representations* using the FFT, and then creating the coefficient representation of the multiplication of two point-value polynomials using the inverse FFT [19]. Because $B(x)$ is the transposed image of $A(x)$, the FFT of $C(x)$ can be computed by transposing the multiplication of the FFT of $A(x)$ by the complex conjugate of the FFT of $A(x)$. For 3D points, we use 3D FFT with the same algorithm as for the 1D case. We note that this is same as computing the cross-power spectrum in the Fourier domain, a technique widely used for image registration [101].

We have implemented our method using FFTW package [34] for computing the

FFT and the inverse FFT, and tested this on several 3D datasets. We are currently using FFTW since we are dealing with quantized values of points that is guided by the precision necessary for the view-dependent rendering application discussed in the next section. After 3D FFT stage, we identify the most common transformations and label them as the *transformation palette*. We currently identify the most common 256 translations and use them in building the vertex-transformation tables and interacting streams of vertices and transformations.

5.6 View-dependent Rendering with Transformation Streams

View-dependent rendering has introduced the concept of rendering different regions of a scene at varying detail based on their perceptual significance. Our view-dependent rendering algorithm is similar to the ones generally used for triangle meshes [77] and points. We first build a binary hierarchy over the input points by following a principal-component-analysis (PCA)-based partitioning [27]. The PCA of a set of n points in a 3D space gives us the mean μ, an orthogonal frame f, and the standard deviation σ of the data. The terms μ and σ are 3D vectors and we refer to their i-th component as μ^i and σ^i respectively, where $\sigma^i \geq \sigma^j$ if $i > j$. The frame f consists of three 3D vectors with the i-th vector referred to as f^i.

The *distortion* of a partitioning is defined as the sum of the distances of the points from the partition's mean [27]. In our partitioning scheme we reduce this distortion by using k-means clustering with $k = 2$. We initialize the two starting means (centers) for the k-means algorithm by doing a PCA over the points and choosing $\mu_p + \frac{\sigma_p^1}{2} f_p^1$ and

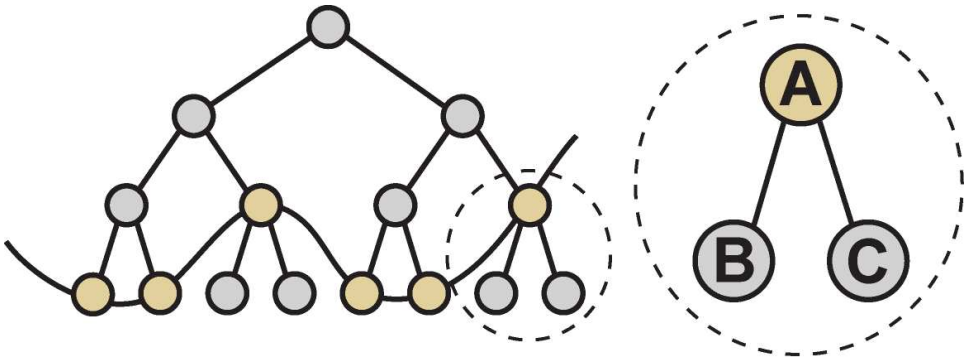


Figure 5.8: *View Dependent Rendering. The entire tree is a complete binary tree. Red nodes were cut in view-dependent manner. Node 'A' can be expanded into two nodes, 'B' and 'C' for the fine details.*

$\mu_p - \frac{\sigma_p^1}{2} f_p^1$ as the initial guesses. This is a reasonable assumption since the data varies maximally along f_p^1. The k-means clustering algorithm then iterates over the twin steps of partitioning the point set according to the proximity of each point to the two means and then updating the two means according to this partitioning. [93] use a geometric way to separate the point set for their point-based simplification hierarchy. They separate along the principal direction f_p^1 with the separating plane passing through the mean μ_p. Their approach is equivalent to the first iteration of our clustering scheme. Subsequent iterations then successively reduce the distortion. We stop iterating when the difference in the distortion between two successive iterations is less than 10^{-7} or when the number of iterations is more than 30, whichever happens earlier.

Next, for each node in the binary tree we carry out the steps mentioned earlier in the dissertation – we identify the most common transformations appropriate for that node, we build vertex-transformation pools, and identify the transformation vertex streams. At the end of this pre-processing step we store the transformation vertex streams with each node in the binary hierarchy.

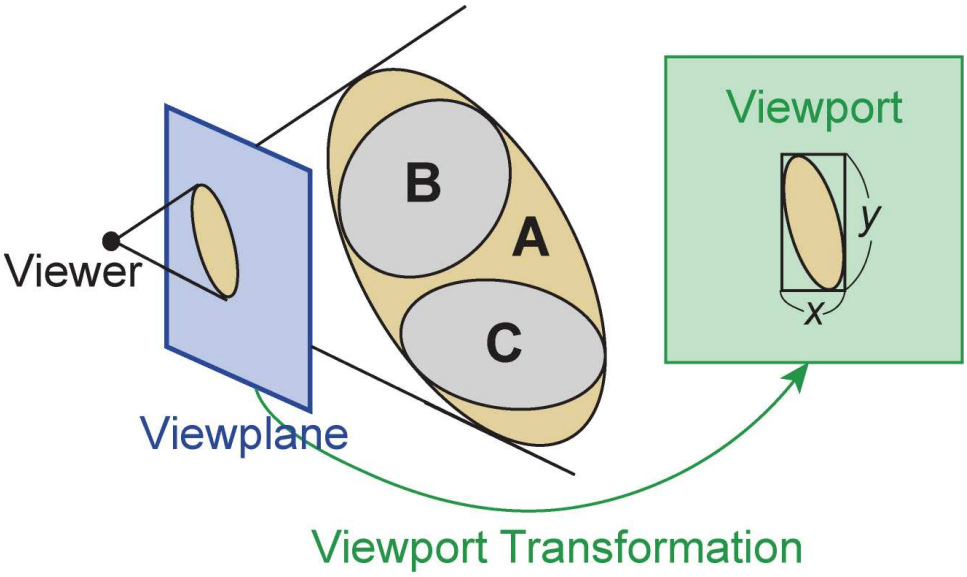


Figure 5.9: *Expansion of a node. Consider the node A in Figure 5.8. We compute how many pixels the bounding box of the node A occupies when it is projected to the window and use it to decide whether to split to its children B and C or not.*

At run time we maintain an active list of nodes representing a cut in this binary hierarchy as shown in Figure 5.8. The points from the active nodes comprise a level of detail appropriate to a given view. We merge sibling nodes if they project to a screen-space area below a threshold and split a parent node into its children if the projected screen-space area is too large. We use the resolution of the projected screen-space area of a node to guide the quantization of the vertices in that node. Thus if a node projects onto a screen-space area no larger than $\frac{128}{\sqrt{2}} \times \frac{128}{\sqrt{2}}$ pixels, a 7-bit vertex quantization along each of x, y, and z axes suffices. While rendering we transform these 7-bit quantized values using the μ, f, and σ values for the node to locate them in the appropriate 3-space.

For each frame we sequentially visit each of the active nodes and decide whether for the given view parameters it will be appropriate to render it directly, or merge it with its siblings, or refine it to its children. Once an appropriate level of detail for a node

has been finalized, we use the vertex transformation streams associated with that node to render the points contained in that node.

5.7 Results

We have validated the results of our approach to efficiently identify and use interacting streams of vertex and transformation data on a number of 3D graphics models. We have run our experiments on a 1.6 GHz Pentium IV Windows PC with 2 GB RAM with a NVIDIA GeForce 6800 Ultra AGP graphics card. We have used the geometry instancing hardware feature in vertex shader 3.0 model and used `DrawIndexedPrimitive( )` command in DirectX 9.0 API.

We have compared our results along two dimensions of performance – the improvement in CPU-GPU communication bandwidth and the improvement in the frame rates. We have measured the frame rates under a constant GPU memory usage model. In this model we allocate a fixed amount of memory on the GPU for rendering using our transformation streams method and using the conventional point rendering method.

Let us assume there are n vertices and m translations in a vertex transformation pool and f is the fraction of the vertices that are actually displayed from the pool (the fraction of unique vertices covered by the pool without redundancy). Then, the pool covers $f \times n \times m$ vertices. Assume we need A_v bytes for a vertex and A_t bytes for a translation in our transformation streams model, and A_n bytes for a vertex in the traditional point rendering model. Our transformation streams model will therefore use $(n \times A_v + m \times A_t)$ bytes instead of the traditional model's $(f \times n \times m \times A_n)$ bytes.

In our experiments $A_v = 26$, $A_t = 12$, and $A_n = 8$. Here A_v is the sum of the bytes for indexing (2 bytes), the number of bytes for a vertex coordinates (8 bytes), and the number of bytes for vertex tagging (16 bytes). A_t is the sum of the number of bytes for translation coordinates (8 bytes) and the number of bytes for translation tagging (4 bytes). `SHORT4` data type is the most compact representation we can use in DirectX to represent vertices and translations, even though we only needed 3 shorts (6 bytes). For a fair comparison between our transformation streams model and the traditional model, we did not exploit the extra short for tagging purposes. The reason we need so many bytes for tagging purposes is because the current-generation GPUs do not support bit-operations for integers in vertex shaders. Therefore, in our experiments we limited ourselves to four floating-point tags per vertex. In each 4-byte floating-point tag we used 22 bits of the mantissa as a fixed-point integer. This allowed us to represent up to 88 translations in a vertex-transformation pool. We decided to limit ourselves to four floating-point vertex tags because of two reasons. First, increasing the number of floats costs more in bandwidth to the vertex shader. Second, using more than four floats required us to differentiate amongst multiple inputs to the vertex shader, thereby adding one more conditional branch in addition to the one shown in Figure 5.5. If in future we are allowed to pass 32-bit unsigned integers directly to vertex shaders we can save 4 bytes for A_v while at the same time increase the number of translations covered to 96. In Section 5.3, we had proposed using 16-bit translation tags. The reason why we instead use two shorts (8 bytes) in our implementation is that we currently pass the remainder and the quotient for the modulo operation in the vertex shader (Figure 5.5) to reduce the number of floor and divide operations.

To give you a flavor of our results, our approach achieved $n = 150$, $m = 88$ (the limit

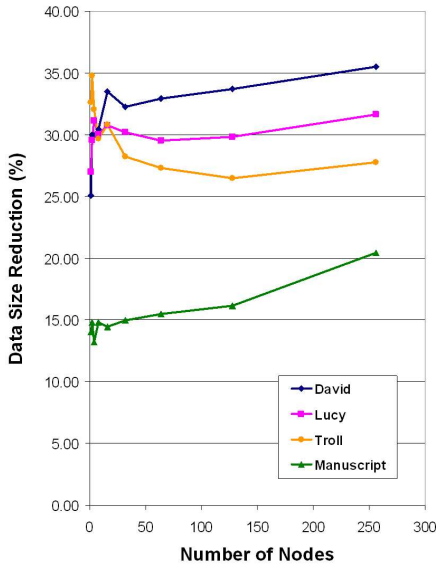


Figure 5.10: *Communication bandwidth improvement for CPU-GPU transmission for transformation streams model compared to traditional point rendering.*

due to four floats discussed above), and $f = 0.32$ for Stanford's David model. Figure 5.10 shows the data transmission gains from using transformation streams as we increase the number of nodes in the hierarchy. We used a 8-level hierarchy for view-dependent rendering. The gains shown include the overhead of sending singleton vertices that our transformation streams model could not cover.

For comparison of frame rates between the transformation streams model and the traditional point rendering model, we used a fixed amount of GPU memory. Let A_{pools} be the amount of data used by the vertex transformation pools in the transformation streams model. For both models we draw A_{pools} amount of data from GPU vertex buffers and the rest from conventional memory.

Our method currently processes geometry without appearance attributes. An easy way to draw models with appearance attributes using our method is to use texturing. The use of texturing results in only a 10% overhead with our method. Visual results of our

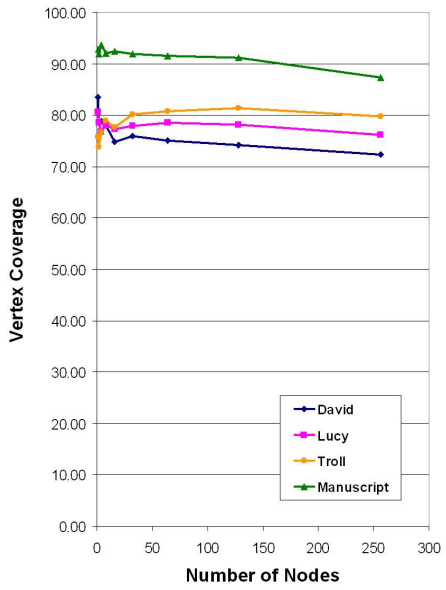


Figure 5.11: *Coverage of vertices by transformation streams model when using two vertex transformation pools.*

approach are shown in Figure 5.12. We have achieved 200% to 500% improvement in communication bandwidth to the GPU and 17% to 32% improvement in frame rates. The left and the center columns of Figure 5.12 shows the conventional rendering and the rendering by Transformation Streams, respectively. As shown in Figure 5.11, two vertex transformation pools can cover about 80% of all the vertices. The right column of Figure 5.12 shows the rendering of the vertices covered by the vertex-transformation pools.

5.8 Conclusions and Future Work

We have developed a novel method for representing 3D point data using interacting streams of vertices and transformations. We have validated this method for accelerating conventional view-dependent rendering applications for points.

Although our method achieves a factor of two to five improvement in the communication requirements to the GPU, our frame rates do not improve by a similar factor. As the graphics community engages in image synthesis for ever larger 3D point datasets and as the gap between processing speeds and memory access times grows ever wider, the impact of our method on graphics rendering performance should rise even further.

One of the important considerations in our method is the space required to encode the boolean interaction matrix. Recent advances in efficiently compressing boolean matrices [54] are relevant to such encodings and suggest a fruitful direction for future research. At present the GPU programmability and the geometry instancing framework offer limited flexibility in exploring sophisticated boolean interaction matrix compression techniques. Still, such compression techniques could greatly assist in remote visualization applications.

We hope that our approach of transformation streams will provide a road-map for future research into how one can use multiple interacting streams in the stream-programming abstraction to map other problems of interest on the GPUs. Our current results appear promising for combining stream programming abstractions with traditional procedural graphics approaches. We believe these are first steps towards more general combinations of multiple data streams for processing geometry, appearance, and physical simulations.

1.69M Verts
20.3FPS / 12.92MB

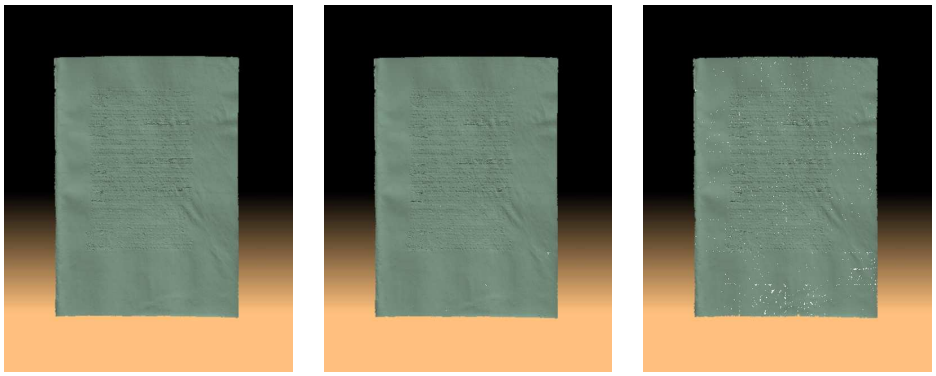
1.69M Verts
24.3FPS / 3.45MB

1.37M Verts
N/A / 997KB

1.17M Verts
29.8FPS / 8.90MB

1.17M Verts
35.0FPS / 2.92MB

878K Verts
N/A / 712KB



1.02M Verts
32.9FPS / 7.75MB
(a) Conventional

1.02M Verts
43.4FPS / 1.25MB
(b) STS

926K Verts
N/A / 565KB
(c) Coverage by pools

Figure 5.12: *The result of rendering XYZ RGB's Troll, Stanford's David and XYZ RGB's Manuscript. The left images show the conventional rendering of the models, the center images show the rendering of them by salient transformation streams and the right images show the vertices covered by the vertex-transformation pools for the models. We report the number of vertices rendering, the frame rates achieved, and the per-frame communication bandwidth required for the conventional approach and our approach.*

Chapter 6

Salient Frame Detection for Molecular Dynamics Simulations

Saliency-based analysis can be applied to time-varying 3D datasets for the purpose of summarization, abstraction, and motion analysis. As the sizes of time-varying datasets continue to grow, it becomes more and more difficult to comprehend vast amounts of data and information in a short period of time. Automatically generated thumbnail images and previewing of time-varying datasets can help viewers explore and understand the datasets significantly faster as well as provide new insights. In this chapter, we introduce a novel method for detecting salient frames for molecular dynamics simulations. Experimental results on E. *coli* mechanosensitive ion channels show the effectiveness of our method.

6.1 Introduction

Recent advances in acquisition and simulation techniques have generated a huge amount of time-varying datasets. Time-varying data can be acquired from scientific simulation, videos, and animation libraries. Features in the time-varying datasets are commonly defined as the regions of interest that a human observer is likely to look for. As the number and complexity of these datasets increase exponentially [53], it is becoming impractical to expect a human observer or a domain expert to discover all the features manually. Automatic or semi-automatic tools to help humans discover scientifically interesting features are especially important for this reason.

Many illustration-based techniques have been proposed by several researchers [5, 56, 123] to summarize time-varying datasets such as ocean flow, volume, and human skeletons. The basic step for these illustration techniques is automatic detection of salient frames which have interesting features. In the method of image saliency by Itti *et al.* [52] or mesh saliency by Lee *et al.* [69], they use a center-surround operator to identify the uniqueness of a pixel or a vertex with respect to its neighborhood. In this chapter, we have decided to use a similar approach and define saliency as the uniqueness of a single frame with respect to its neighboring frames both forwards and backwards in time. Our collaborator, Dr. Sergei Sukharev's group at Biology Department at the University of Maryland, was interested in identifying the frames in molecular dynamics simulations, where the anomalies (kinks) in the secondary structures happen in the opening and closing simulations of the *Ecoli* channel [3]. We validate the effectiveness of our salient frame detection algorithm in this molecular dynamics simulation. To the best of our knowledge, there have been no salient frame detection techniques for molecular dynamics simulations despite a great need for such tools for this area.

The rest of this chapter is organized as follows. A review of related work is provided in Section 6.2. In Section 6.3, we formulate the relationship between one residue and the neighboring residues in space, and present an algorithm to detect saliency in time. Results are presented in Section 6.4, Section 6.5 concludes this chapter and discusses future work.

6.2 Related Work

This section briefly reviews the related research in the areas of (1) saliency analysis for time-varying 3D datasets and (2) some introductory background for protein structures and ion channels.

6.2.1 Saliency-based Motion Analysis

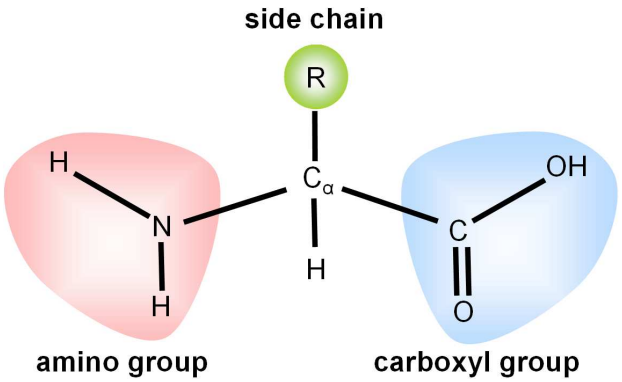
Designers and artists have long used a single static image or a few images to illustrate dynamics of scenes for motion. They have depicted dynamics to facilitate visual communication in comic books and storyboards [82]. Recently, several graphics researchers [56, 87, 97] have proposed illustration-based techniques to depict the dynamics of time-varying data in a compact way. They use principles of visual art such as glyphs, and generate an image (or a few images) to summarize the time-varying data to facilitate visual communication. For instance, Joshi and Rheingans [56] have used illustration-based techniques such as speedlines, flow ribbons, and strobe silhouettes to convey change over time for a time-varying dataset. Nienhaus and Dollner [87] have used dynamic glyphs such as directed acyclic graphs and behavior graphs to provide further information about dynamics in the 3D scene.

A very interesting beginning in detecting salient frames for human skeleton datasets has been made by Assa *et al.* [5]. They generate an action synopsis for presenting the motion of a single skeleton-based character. They represent motion in affinity matrices, constructed from various aspects of a pose such as joint positions, joint velocities, joint angles, and joint angular velocities. They first define a vector $\mathbf{x_a^k}$ which represents an as-

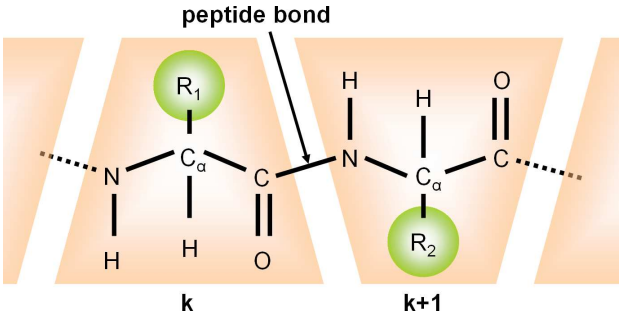
pect a of the pose at frame k. Then, they compute the dissimilarity of the aspect a between two given frames i and j by a simple distance measure to identify key poses. Finally, they compose these key poses into a single image by including the most significant poses.

6.2.2 Protein Structures

A protein structure is formed by a unique three-dimensional assembly of a specific polypeptide chain. Each polypeptide chain contains a particular sequence of serially linked amino acids. Figure 6.1(a) shows an amino acid which is composed of an amino group, a carboxyl group, and a side-chain, which are connected at the central C_α atom.



(a)



(b)

Figure 6.1: *Image (a) shows the structure of an amino acid. Image (b) shows a peptide bond formed by the reaction between a carboxyl group of one amino acid and an amino group of the other amino acid. Each peptide bond releases a molecule of water (H_2O). Images are adapted from [12].*

When the carboxyl group of one amino acid reacts with the amino group of another amino acid, a peptide (i.e., amide) bond (Figure 6.1(b)) is formed by releasing a molecule of water (H_2O). This peptide bond is typically composed of four atoms (C, O, N, and H) which lie on a common plane due to the partial double bond characteristic at the CO-NH connection. Here, the recurring atomic array of N-C_α-C(=O) from each amino acid of a polypeptide chain constitutes the protein *backbone*. By definition, the specific amino acid sequence for each polypeptide chain is the *primary* structure of the protein. Segments of polypeptides often fold locally into stable structures such as α-helices or β-strands, each of which is called a *secondary* structure. Typically, the α-helix is a right-handed coiled conformation, resembling a spring. β-strands connected laterally by three or more hydrogen bonds, form a generally twisted, pleated sheet.

The angle between two planes is referred as their *dihedral* angle. Figure 6.2(a) and (b) shows how we can compute the dihedral angle when there are four atoms which are not co-linear in 3D space. We first align the atoms B and C as shown in Figure 6.2(b). Then the dihedral angle corresponds to the angle measured in clockwise direction between the atom A and the atom D. Similarly, for a sequence of protein's polypeptide chain, backbone atoms (C, N, and C_α) allow for three different dihedral angles of proteins as depicted in Figure 6.2(c): ϕ involving the backbone atoms C-N-C_α-C, ψ involving the backbone atoms N-C_α-C-N, and ω involving the backbone atoms C_α-C-N-C_α. The planarity of the peptide bond usually restricts ω to be 180° or 0°. Thus the Ramachandran plot [100] considers two variable dihedral (torsion) angles (ϕ and ψ) and shows possible combinations of these conformational angles of representative secondary structures in a polypeptide such as α-helices or β-sheets.

6.2.3 Ion Channels

Ion channels are proteins that regulate the flow of ions into and out of the cells. Ion channels enable a very rapid flow of ions – sometimes as many as a million ions a second. Ion channel transitions are very fast – some opening for less than a millisecond before they close. This rapid and highly specific gating of ion channels is necessary for survival of cells. The speed at which ions flow across the cell membrane (i.e., the ion channel kinetics) impacts the reaction time of a nerve or a muscle cell, and thus dictates the response time of the animal to the possible environmental dangers. An accurate understanding of the structural changes and functioning of ion channels is vital for therapeutic drug design. Nearly a third of the top 100 pharmaceutical drugs target ion-channels.

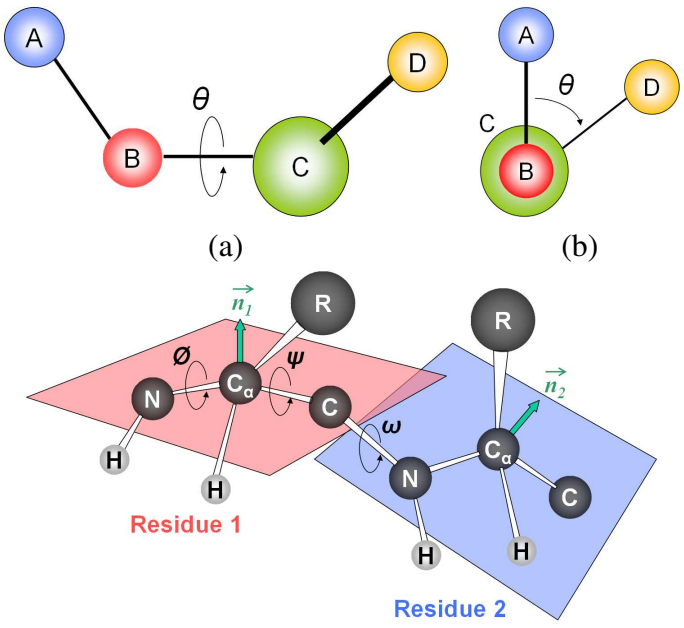
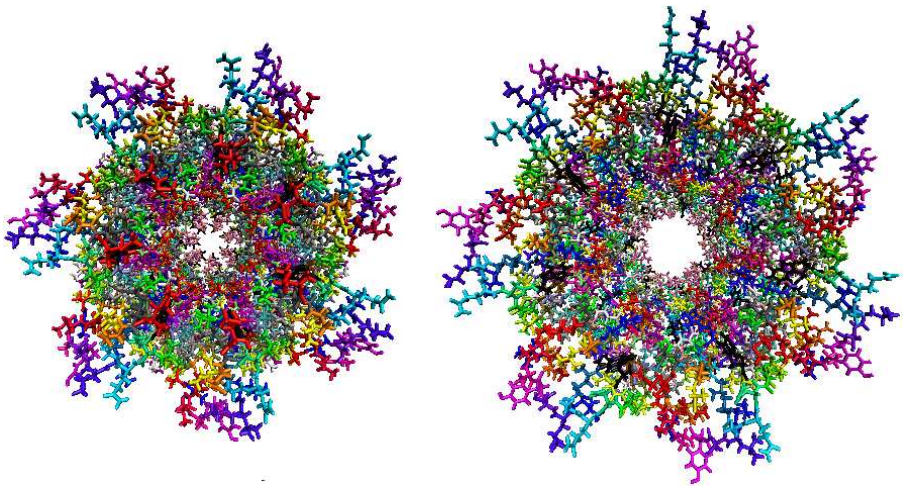


Figure 6.2: *Image (a) and (b) show the computation of a dihedral angle between 4 atoms (A, B, C, and D). When we align the atom B and the atom C as shown in Image (b), the dihedral angle θ is defined as the angle between the atom A and the atom D in clockwise direction. Image (c) shows the dihedral angles (ϕ between C-N-C_α-C, ψ between N-C_α-C-N, and ω between C_α-C-N-C_α) and the normal vectors ($\vec{n}_1$ and $\vec{n}_2$ on the planes defined using N-C_α-C in residue 1 and residue 2, respectively. Images are adapted from [12].*



(a) (b)

Figure 6.3: *Image (a) and (b) show opening of the E.* coli *mechanosensitive ion channel, respectively. There are seven subunits in this ion channel, and all seven subunits are topologically identical, but do not act independently in the simulation.*

6.3 Salient Frame Detection

In this chapter, we define saliency as the uniqueness of a single frame with respect to its surrounding frames in time, and detect the salient frames for molecular dynamics simulations. Mechanosensitive ion channels play a critical role in transducing physical stresses at the cell membrane into an electrochemical response. The crystal structure of E. *coli* MscS has provided a starting point for detailed descriptions of its mechanism. Figure 6.3 shows the opening of the E. *coli* mechanosensitive ion channel that we will consider throughout this chapter. There are 7 subunits in this ion channel, and all 7 subunits are topologically identical, but do not act independently in the simulation. Each subunit has residues 1 to 175 (with few gaps cut out). To understand this mechanism, identifying the presence of kinks in α-helices is critical because they have functional importance. Kink detection, however, is a tricky question because there are many factors involved. These include the state (ruptured or not) of the H bonds, local geometric infor-

mation such as Ramachandran angles (torsion angles), and more global information such as the angles among multiple atoms.

In this section, we formulate the relationship between one residue and the neighboring residues spacially, and present an algorithm to detect saliency in time. Our framework encompasses the global and local geometric properties of backbone residues in a molecular dynamics simulation.

6.3.1 Construction of Affinity Matrices in Space

We explore the relationship between one residue and the neighboring residues to detect the changes in α-helices. The straightening and buckling of α-helices are interesting because they appear in many of simulations of ion channels and are believed to be correlated with conformational states of the whole channel [115]. There are many ways to define the relationship among residues, but we believe the angles in backbone atoms would be one of the best ways since backbone atoms are much more stable in their positions than side-chains. As Ramachandran plot suggests, we could have measured torsion (dihedral) angles and conjectured the changes of secondary structures for each residue. However, analysis of Ramachandran angles only considers very local properties inside a residue, and does not encompass the global geometric property among a sequence of residues. Instead, we use the relative angles between one C_α (α-carbon) and other α-carbons within a cut-off distance r_s.

Molecular dynamics simulation gives us a trajectory file which holds all the atom positions in 3D space for every frame k. Since three non-colinear points in 3D space

can define a plane, the positions of N-C_α-C atoms in each residue can define a plane and its normal vector $\vec{n}$ as shown in Figure 6.2(c). We first compute normal vectors ($\vec{n}_i$) to the planes formed by these N-C_α-C atoms in residues (R_i) for every frame k. Then, we construct an affinity matrix A_k for the k-th frame as:

$$A_k = \begin{bmatrix} \vec{n_1}\cdot\vec{n_1} & \vec{n_1}\cdot\vec{n_2} & \vec{n_1}\cdot\vec{n_3} & \dots & \vec{n_1}\cdot\vec{n_m} \\ \vec{n_2}\cdot\vec{n_1} & \vec{n_2}\cdot\vec{n_2} & \vec{n_2}\cdot\vec{n_3} & \dots & \vec{n_2}\cdot\vec{n_m} \\ \vec{n_3}\cdot\vec{n_1} & \vec{n_3}\cdot\vec{n_2} & \vec{n_3}\cdot\vec{n_3} & \dots & \vec{n_3}\cdot\vec{n_m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vec{n_m}\cdot\vec{n_1} & \vec{n_m}\cdot\vec{n_2} & \vec{n_m}\cdot\vec{n_3} & \dots & \vec{n_m}\cdot\vec{n_m} \end{bmatrix} \tag{6.1}$$

where m is the number of residues that we consider. Here each element $a_k(i,j)$ is simply a dot product between the normals $\vec{n}_i$ and $\vec{n}_j$. If the difference between two residues R_i and R_j is larger than a cut-off distance r_s, we set $a_k(i,j)$ to be zero. On average, there are 3.6 amino acids per turn in α-helices. Throughout this chapter, we use $r_s = 5$, which covers about 3 turns forwards and backwards in α-helices. Figure 6.4 visualizes the affinity matrices from the first and the second frames for 33 residues (from residue 94 to residue 126) of the subunit 1 for the molecule shown in Figure 6.3.

6.3.2 Saliency Detection among Neighboring Affinity Matrices

Our affinity matrix in equation 6.1 represents geometric relationship among neighboring residues. This angular relationship cannot be represented by a single vector $\mathbf{x}$ as in [5]. Therefore, the dissimilarity between two given frames i and j should be computed

by the difference between two affinity matrices A_i and A_j. There are several ways to compute the difference between two matrices. However, we have decided to use singular value decomposition (SVD) in computing the difference between two matrices. Singular value decomposition (SVD) [39] factorizes a given $m \times n$ matrix A into three matrices: $A = U \times \Sigma \times V'$, where U is an $m \times m$ unitary matrix ($U \times U' = I$ and $U' \times U = I$), Σ is $m \times n$ diagonal matrix with nonnegative numbers, and V' is the transpose of an $n \times n$ unitary matrix V. There are two nice properties in the SVD decomposition: (1) U and V are a set of orthonormal basis vectors (singular vectors), and (2) the diagonal entries in Σ are called singular values, which are sorted in non-increasing order and indicate the importance of the corresponding basis vectors. If there is noise in the original matrix A, the noise is represented by the least important basis vectors and singular values.

In α-helices, backbone atoms (C, O, N, and H) are much more stable than side-chains because of the H bonds. However, there could be still a lot of vibrations in the positions of backbone atoms over time. By using SVD analysis, we expect these vibra-

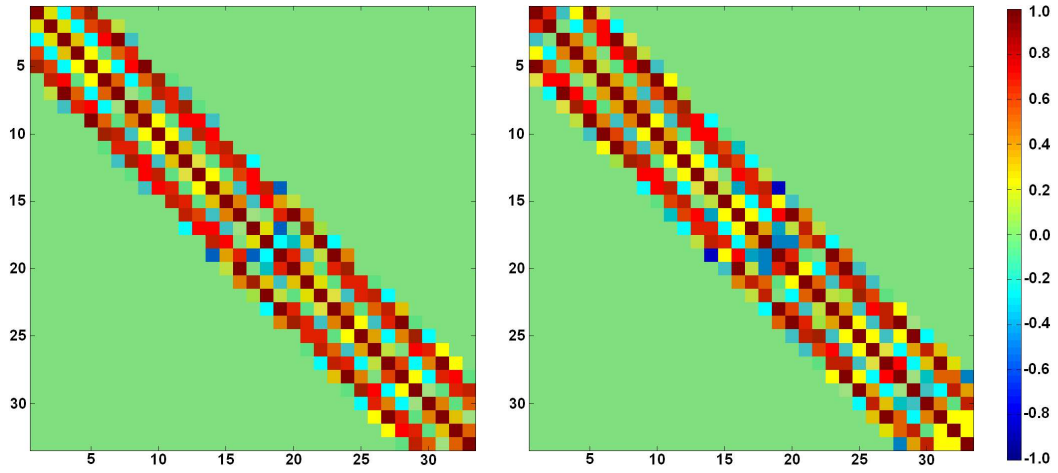


Figure 6.4: *Visualization of affinity matrices computed from the first and the second frames for* 33 *residues of the subunit* 1 *for E.* coli *MscS (shown in Figure 6.3) when the cut-off distance,* $r_s = 5$ *is used.*

tions to occur amongst the least significant singular values, which are ignored. This is why we use SVD over any other methods for computing the uniqueness of the affinity matrix A_i from the affinity matrix A_j.

Uniqueness of the affinity matrix A_i from the affinity matrix A_j: We perform a singular value decomposition on A_i: $A_i = U_i \times \Sigma_i \times V_i'$. This returns the best basis vectors as the column vectors in U_i. Since the basis vectors are sorted by their importance in SVD decomposition, we can obtain a reduced matrix $\hat{U}_i$ by taking the first r_i basis vectors in U_i. For the j-th frame A_j, we use these r_i basis vectors to best approximate it. For this, we project A_j to the low-dimensional subspace spanned by the r_i basis vectors as: $W_{i,j} = \hat{U}_i' \times A_j$. This gives us the weight matrix $W_{i,j}$ for the r_i basis vectors. We use this weight matrix to approximate A_j by: $\hat{A}_j = \hat{U}_i \times W_{i,j}$. Finally we compute the root mean square error (ε_{ij}) between $\hat{A}_j$ and A_j: $\| \hat{A}_j - A_j \|_F$. , where the Frobenius norm of a matrix M is defined as $\| M \|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |l_{ij}|^2}$.

Saliency Value s_i for the frame i: To compute the uniqueness of a frame i from other frames j, we avoid considering all possible pairs (i, j). Instead, we consider neighboring frames j where $|i - j| \leq r_t$. Throughout this chapter, we use $r_t = F/10$, where F is the number of total frames. The final saliency value s_i is the average of the errors ε_{ij} in neighboring frames of i:

$$s_i = \frac{\sum_{|j-i| \leq r_t} \varepsilon_{ij}}{F_i}$$

where F_i is the number of frames whose distance from the frame i is less than or equal to r_t. Figure 6.5 shows the graph for these saliency values in blue.

6.4 Results

We have compared our detected salient frames with the ones identified independently by our collaborators (biology scientists) for molecular dynamics simulations. Figure 6.5 shows the five most salient frames detected by our method for the subunit 4 in the E. *coli* mechanosensitive ion channel in Figure 6.3. The frames 5, 26, 30, and 34 which have been detected by our method are the same or very close to the frames 3, 24, 26, 30, and 35 with changes in the kinks, which were detected manually by our collaborators. The frame 39 detected by our algorithm is not close to any frame detected manually by our collaborators, but it had the lowest saliency value among the five most salient frames. Generally, kinks change towards the end of this simulation, and our method successfully detects these important frames.

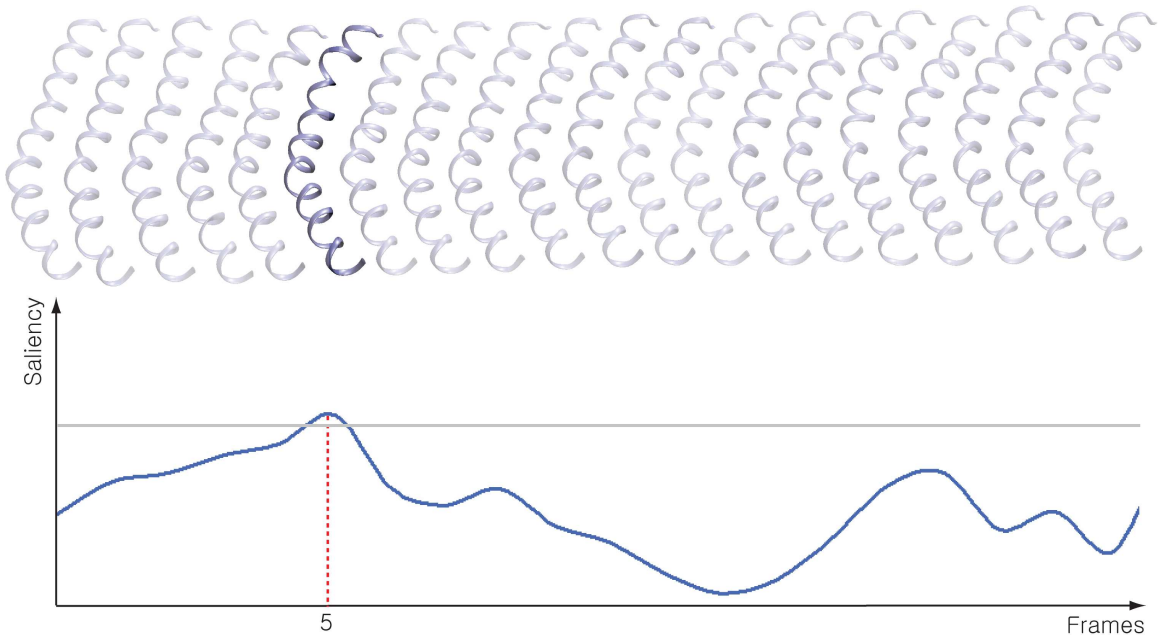
Figure 6.6 shows the five most salient frames detected by our method for the subunit 1 in the ion channel shown in Figure 6.3. This subunit is topologically identical to the subunit 4, but acts differently in the simulation. Therefore, it results in different salient frames (frames 11, 19, 21, 35, and 39) as shown in Figure 6.6. Our collaborators identified frames 2, 18, 20, 23, 35, 36, and 39 as being salient. Figure 6.7 shows the six most salient frames detected by our method for the subunit 4 in the symmetry annealing of MscS F68S mutant. In this molecular dynamics simulation, residue 68 was mutated to another, serine, which has very specific consequences for channel inactivation in real experiments. As changes in the kinks occur more frequently than the previous simulations, we observe a larger number of salient frames than in the previous cases. Our collaborators have manually identified frames 2, 4, 18, 34, and 38 as being salient. Among these, frames

2, 4, 18, and 38 are the same or close to the frames 1, 5, 18, and 39 detected by our algorithm, and the remaining frame 34 also exhibits a relatively high saliency value as shown in Figure 6.7.

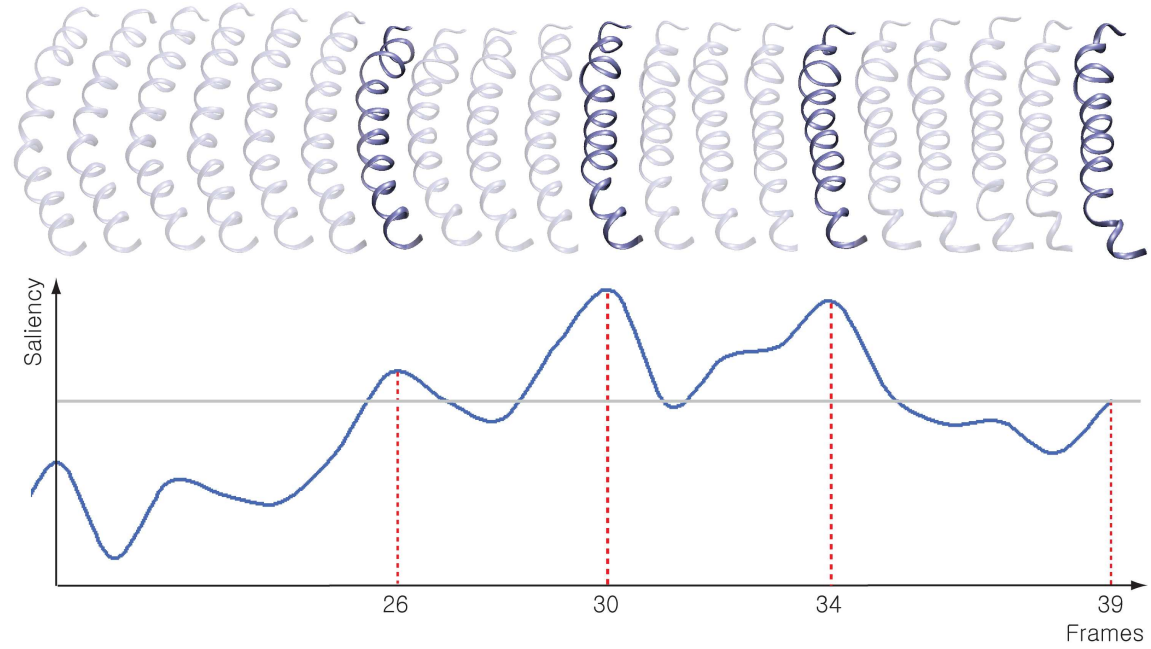
6.5 Conclusions and Future Work

In this chapter, we have detected salient frames for molecular dynamics simulations. We have introduced the notion of saliency in time, and successfully identified some of the key frames which have changes in the kinks (i.e., appearance or disappearance of a kink) for *Ecoli* channel. We believe that our method can enable researchers to focus on the important frames for further analysis of the dataset.

We currently consider the angles among residues in α-helices, and identify the anomalies (kinks) in the secondary structures for an *Ecoli* channel. We believe, however, this framework can be easily extended to encompass other salient features in molecular dynamics simulations by changing the way we construct affinity matrices to address other needs by scientists or domain experts. In this context, it will be interesting to compare and contrast the results of salient frames detected by a new method with what scientists think salient in their domains.

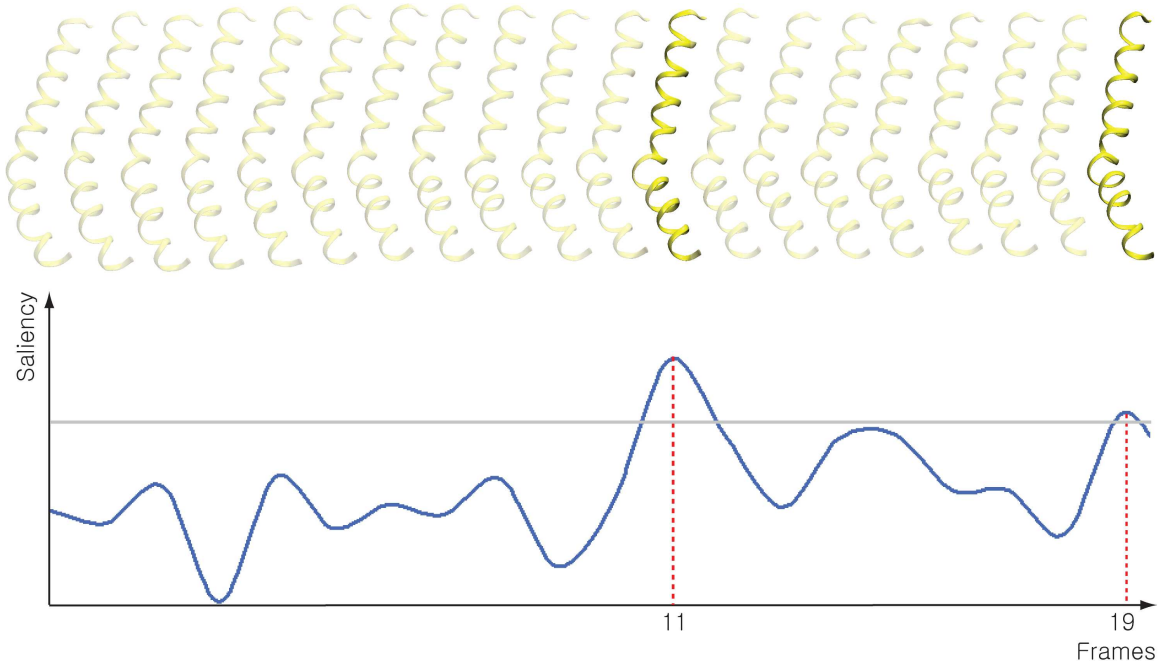


(a) Frames 0 to 19

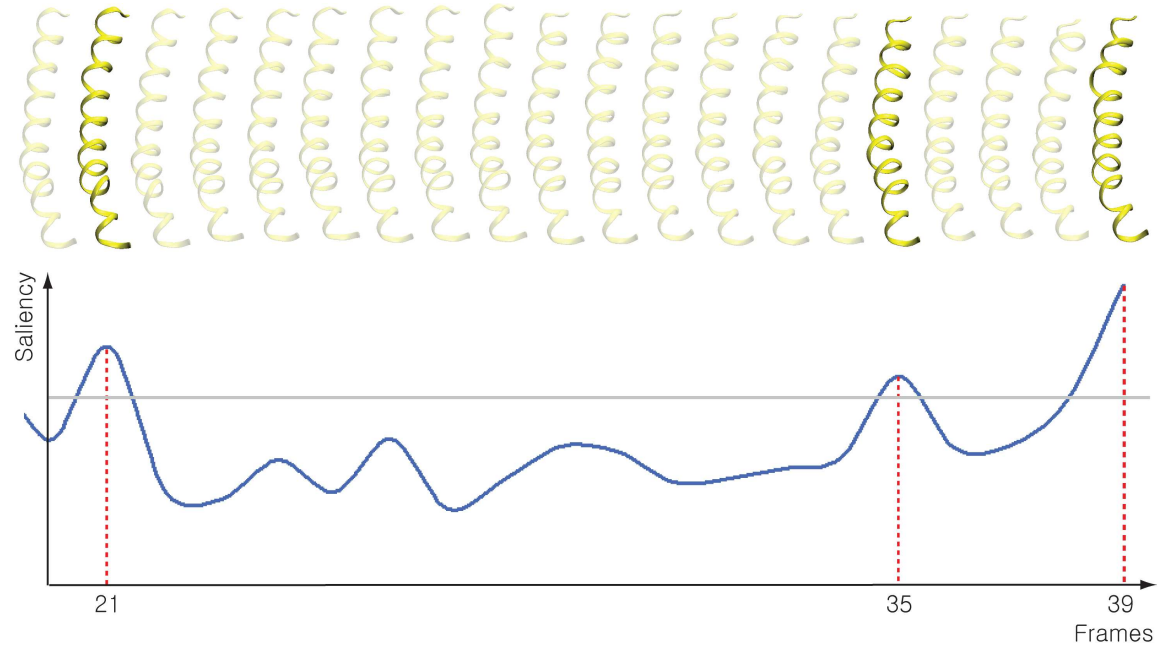


(b) Frames 20 to 39

Figure 6.5: *Five most salient frames detected by our method for the subunit* 4 *in the E.* coli *ion channel (MscS) in Figure 6.3. The changes in the kinks are detected towards the end of this simulation, and our method successfully detects some of the most important frames.*

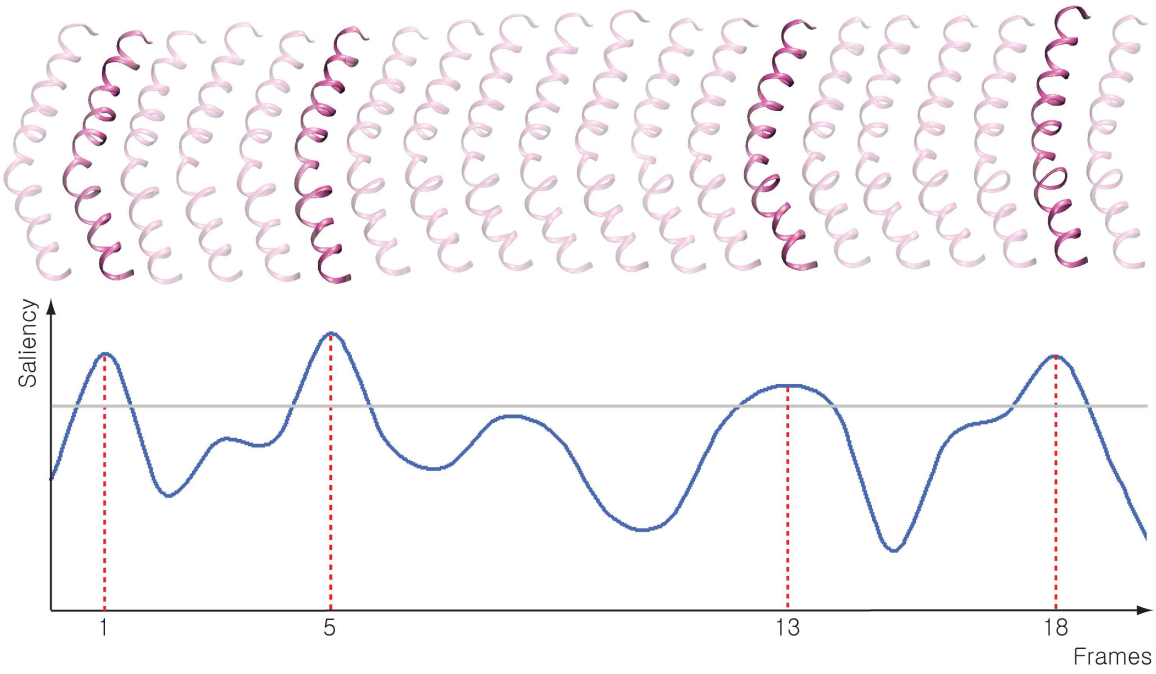


(a) Frames 0 to 19

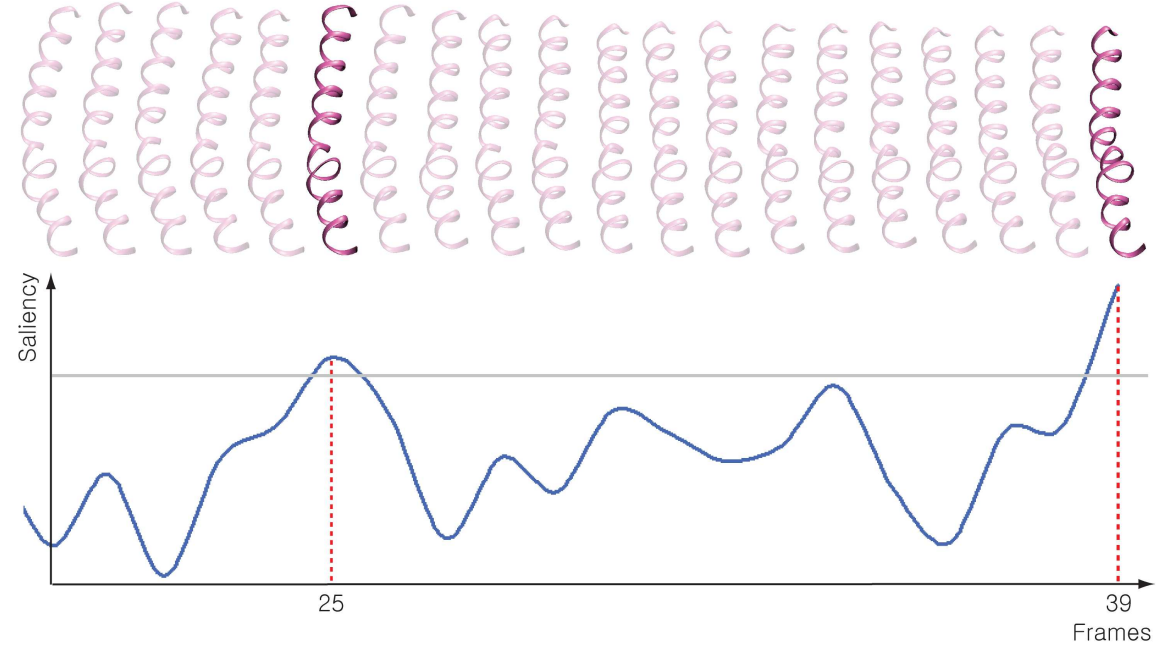


(b) Frames 20 to 39

Figure 6.6: *Five most salient frames detected by our method for the subunit* 1 *in the E.* coli *ion channel (MscS) in Figure 6.3. This subunit is topologically identical to the subunit* 1*, but acts differently in the simulation.*



(a) Frames 0 to 19



(b) Frames 20 to 39

Figure 6.7: *Six most salient frames detected by our method for the subunit* 4 *in the other molecular dynamics simulation, showing the symmetry annealing of MscS F68S mutant – the residue* 68 *was mutated to another, serine, which has very specific consequences for channel inactivation in real experiments.*

Chapter 7

Conclusions and Future Work

Our ability to generate 3D data has far surpassed our ability to visually comprehend it due to the recent advances in acquisition and simulation technologies. Therefore, graphics and visualization applications need to incorporate principles of visual perception to allow people to comprehend information more quickly. Our research has focused on having a validated model of mesh saliency, using a saliency-guided operator to enhance desired regions of 3D datasets, and identifying salient frames for time-varying datasets.

Our research has shown that the previous model of mesh saliency can well approximate human eye movements. We believe looking for non-repeating structure in the middle of repeating structure is another way to identify the salient regions. We have shown that this high-level representation to extract important components of the data set is useful not only for analyzing but also for improving the interactivity of graphics applications. Next, we have shown how one can use saliency for enhancing specific region of interest both in a 3D mesh and in a volume. Our saliency-guided enhancement operator is more effective than the conventional Gaussian-based enhancement in drawing viewers' attention to a desired region of interest. Finally, we have shown that the notion of saliency can be applied to time-varying 3D datasets. Our SVD-based analysis on a time-varying dataset has successfully identified salient frames in molecular dynamics simulations. During our studies, we have uncovered several interesting issues that need further investigation, such as con-

sidering multiple appearance attributes in the computational model of mesh saliency and applying our enhancement operator to arbitrary regions of interest with different levels of enhancement. Finally, we believe that our salient frame detection framework can be also useful for anomaly detection in large time-varying datasets.

7.1 Multichannel Mesh Saliency

Previous models of mesh saliency have only considered geometric attributes such as mean curvature and its variation. However, visualization systems today are characterized by a rich visual complexity that arises from several visual channels including color, texture, and geometry. Figure 7.1(a) shows a Stanford Bunny model with high variance texture. There are many regions where the geometry has little variation, but the color intensity and the color opponency have large variation. Generalizing mesh saliency to encompass these attributes should be an important direction for further research. Computing each channel of saliency for a 3D model does not seem complicated if we want to use the same multi-scale center-surround mechanism proposed in Lee *et. al.* [69]. We can just change the scalar values from mean curvature to other property such as color intensity and color opponency for each channel. Figure 7.1(b), (c), and (d) show multichannel saliency values for the Stanford Bunny Model in Figure 7.1(a).

The real challenge in computing multichannel mesh saliency is how to aggregate these channels. We should be able to answer the following questions. Is a region of a mesh that has a high color contrast but a low curvature gradient more effective at eliciting viewer attention than another region that has a high curvature gradient but a lower color

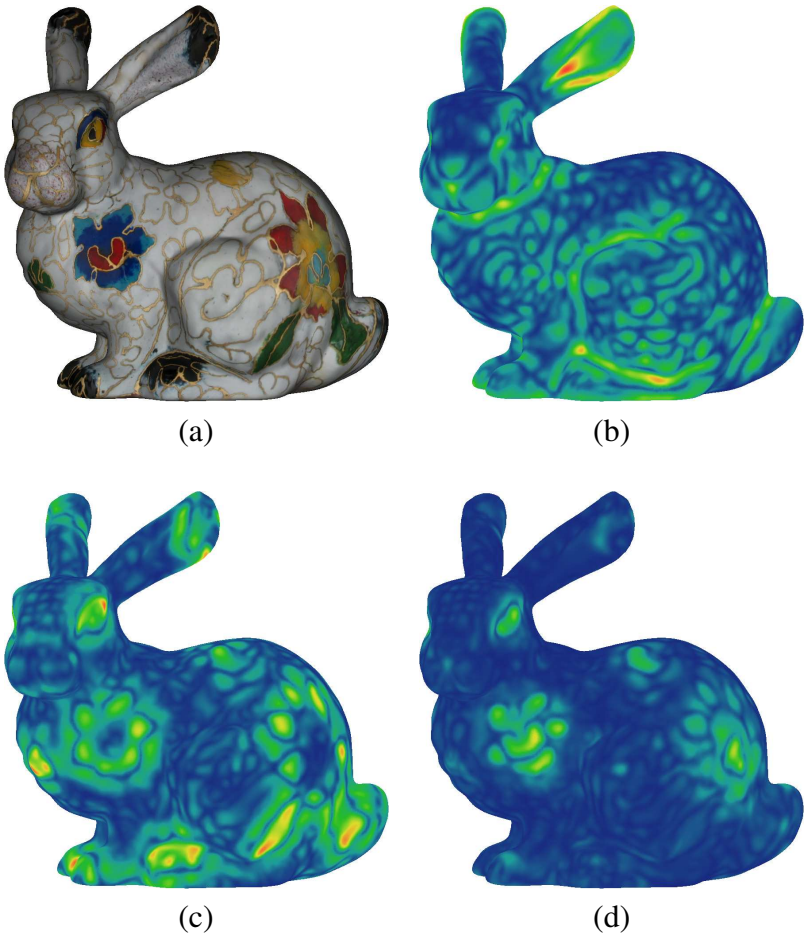


Figure 7.1: *Multichannel Saliency Values for the Stanford Bunny Model. Image (a) shows the Stanford Bunny model with high variance texture. Image (b), (c), and (d) show the saliency from the gradient of mean curvatures, the gradient of color intensity, and the color opponency by the normalized sum of the saliency at all scales, respectively.*

opponency? Are some visual channels clearly superior in eliciting visual attention than others? Should we use the same scale or scale weights for different salient channels? Finally, the new multichannel mesh saliency model should be validated again with respect to human eye movements through eye-tracking-based user study.

7.2 General Saliency Detection in Large Time-Varying Datasets

Our salient frame detection framework is extremely useful for detecting other anomalies in these large time-varying 3D datasets. Right now, we construct a matrix for each frame i and do the SVD analysis on it to compute the most important basis vectors. We then use these basis vectors from the frame i to approximate neighboring frames j. This results in the approximation errors, which are the saliency values in our salient frame detection framework in Chapter 6.

7.2.1 Data Analysis for 4D Spaces

As the size of time-varying 3D datasets grows, there have been several efficient visualization techniques developed based on parallel algorithms for graphics processing unit (GPU) or multi-core processors [103, 110, 126]. Large time-varying 3D datasets can be regarded as 4D datasets. We can also carry out our analysis on any combination of space and time. For instance in molecular dynamics simulations, if we construct a matrix for each residue with the information of all the time frames, we can do analysis similar to what we have presented in Chapter 6 to detect anomalous movements at the residue level.

7.2.2 Animation Data Compression

As the computing power has been increasing, one of the greatest bottlenecks in rendering 3D datasets is the data transfer time from the memory to the graphics processor. Our salient frame detection framework can be adapted to progressively update motion subspaces for fast rendering of large time-varying datasets. Instead of formulating a ma-

trix for each frame, we could construct one matrix for all frames. The k-th column of this matrix holds the information of the frame k. Then, we can find a fixed set of motion subspaces which can be shared by consecutive frames. We automatically update these subspaces only if a new salient frame emerges. This idea can significantly improve the communication bandwidth requirement in rendering animation data.

7.3 Enhancement for Multiple Regions

Our saliency-based enhancement technique for volume datasets has been validated on spherical regions of interest and a binary-valued saliency field. Generating an emphasis field from an arbitrary-shaped region with general saliency values can be considered in the future. With this advance, one can see the effect of our method on two or more regions with varying degrees of importance.

Extending our technique for multiple regions, however, is not obvious. Assume there are two regions of interest with different degrees of importance. This will result in two different emphasis fields around them with our current technique. If they are separated from each other, we can combine them easily. However, if they overlap, there could be several ways to combine them. For instance, we can compute the mean or take the maximum of two field values at each voxel. Instead of trying to combine two emphasis fields in 3D space with our current technique, we can also think of performing the whole computation in frequency domain because the convolution of the Gaussian function and our emphasis field can be easily processed in that space. It will be interesting to investigate several possibilities for aggregating emphasis fields for two or more regions

of interest. Once we have a technique to enhance multiple regions with varying degrees of importance, it should be validated with an eye-tracking-based user study.

Bibliography

[1] M. Agrawala and F. Durand. Smart depiction for visual communication. *IEEE Computer Graphics and Applications*, 25(3):20–21, 2005.

[2] D. Akers, F. Losasso, J. Klingner, M. Agrawala, J. Rick, and P. Hanrahan. Conveying shape and features with image-based relighting. In *Proceedings of IEEE Visualization*, pages 349 – 354, 2003.

[3] B. Akitake, A. Anishkin, N. Liu, and S. Sukharev. Straightening and sequential buckling of the pore-lining helices define the gating cycle of mscs. *Nature Structural and Molecular Biology*, 14(12):1141–1149, 2007.

[4] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, C. Silva, and D. Levin. Point set surfaces. In *IEEE Visualization 2001*, pages 21–28, October 2001.

[5] J. Assa, Y. Caspi, and D. Cohen-Or. Action synopsis: pose selection and illustration. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 24(3):667–676, 2005.

[6] C. Bajaj and G. Xu. Anisotropic diffusion of surfaces and functions on surfaces. *ACM Transactions on Graphics*, 22(1):4–32, 2003.

[7] I. Baran, E. D. Demaine, and M. Patrascu. Subquadratic algorithms for 3SUM. In *Workshop on Algorithms and Data Structures (WADS)*, pages 409–421, 2005.

[8] G. A. Blaauw and F. P. Brooks, Jr. *Computer architecture: concepts and evolution*. Addison-Wesley, Reading, MA, USA, 1997.

[9] J. Bloomenthal, C. Bajaj, J. Blinn, M.-P. Cani-Gascuel, A. Rockwood, B. Wyvill, and G. Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufmann, 1997.

[10] M. Botsch and L. Kobbelt. High-quality point-based rendering on modern GPUs. In *Pacific Conference on Computer Graphics and Applications*, pages 335–343, 2003.

[11] M. Botsch, A. Wiratanaya, and L. Kobbelt. Efficient high quality rendering of point sampled geometry. In *Proceedings of the 13th Eurographics Workshop on Rendering*, pages 53–64. Eurographics Association, 2002.

[12] C. Branden and J. Tooze. *Introduction to Protein Structure*. Garland Publishing, Inc., second edition, 1999.

[13] S. Bruckner, S. Grimm, A. Kanitsar, and E. Gröller. Illustrative context-preserving volume rendering. In *Proceedings of EuroVis 2005*, pages 69–76, May 2005.

[14] I. Buck, T. Foley, D. Horn, J. Sugerman K. Fatahalian, M. Houston, and P. Hanrahan. Brook for GPUs: stream computing on graphics hardware. *ACM Transactions on Graphics*, 23(3):777–786, August 2004.

[15] M. Burns, J. Klawe, S. Rusinkiewicz, A. Finkelstein, and D. DeCarlo. Line drawings from volume data. *ACM Trans. on Graphics (SIGGRAPH 2005)*, 24(3), August 2005.

[16] B. Cohen. *Explaining Psychological Statistics*. John Wiley & Sons, Inc., second edition, 2001.

[17] D. Cohen-Or, Y. L. Chrysanthou, C. T. Silva, and F. Durand. A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):412–431, 2003.

[18] F. Cole, D. DeCarlo, A. Finkelstein, K. Kin, K. Morley, and A. Santella. Directing gaze in 3D models with stylized focus. In *Eurographics Workshop/ Symposium on*

Rendering, pages 377–387, 2006.

[19] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1996.

[20] C. Dachsbacher, C. Vogelgsang, and M. Stamminger. Sequential point trees. *ACM Transactions on Graphics*, 22(3):657–662, 2003.

[21] B. Dally. Stream processors vs. GPUs. In *GP2: 2004 ACM Workshop on General-Purpose Computing on Graphics Processors*, pages A–13, 2004.

[22] W. J. Dally, P. Hanrahan, M. Erez, T. J. Knight, F. Labonte, J. Ahn, N. Jayasena, U. J. Kapasi, A. Das, J. Gummaraju, and I. Buck. Merrimac: Supercomputing with streams. In *Supercomputing*, page 35, 2003.

[23] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. *ACM Transactions on Graphics*, 22(3):848–855, 2003.

[24] D. DeCarlo and A. Santella. Stylization and abstraction of photographs. *ACM Trans. on Graphics (SIGGRAPH 2002)*, 21(3):769–776, 2002.

[25] M. Desbrun, M. Meyer, P. Schröder, and A. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of ACM SIGGRAPH*, pages 317–324, 1999.

[26] M. Desbrun, M. Meyer, P. Schröder, and A. Barr. Anisotropic feature-preserving denoising of height fields and bivariate data. In *Proceedings of Graphics Interface*, pages 145–152, 2000.

[27] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., New York, second edition, 2001.

[28] B. Dudash. Mesh instancing. Technical report, NVIDIA Corporation, 2701 San

Tomas Expressway, Santa Clara, CA 95050, 2004.

[29] N. Dyn, D. Levin, and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, 1990.

[30] J.T. Enns and R.A. Rensink. Sensitivity to three-dimensional orientation in visual search. *Psychological Science*, 1(5):323–326, 1990.

[31] M. Feixas, M. Sbert, and F. Gonzalez. A unified information-theoretic framework for viewpoint selection and mesh saliency. *ACM Transactions on Applied Perception*, page (to appear), 2008.

[32] S. Fleishman, D. Cohen-Or, M. Alexa, and C. T. Silva. Progressive point set surfaces. *ACM Transactions on Graphics*, 22(4):997–1011, 2003.

[33] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. *ACM Trans. on Graphics (Procs. ACM SIGGRAPH)*, 22(3):950–953, 2003.

[34] M. Frigo and S. G. Johnson. FFTW: An adaptive software architecture for the FFT. In *ICASSP conference proceedings*, volume 3, pages 1381–1384, 1998.

[35] A. Gajentaan and M. H. Overmars. On a class of O(n^2) problems in computational geometry. *Computational Geometry: Theory and Applications*, 5:165–185, 1995.

[36] R. Gal and D. Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics*, 25(1):130–150, 2006.

[37] J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Trans. Graph.*, 23(1):45–63, 2004.

[38] J. P. Grossman and William J. Dally. Point sample rendering. In *Rendering Techniques '98*, Eurographics, pages 181–192. Springer-Verlag Wien New York, 1998.

[39] M. Gu and S.C. Eisenstat. Downdating the singular value decomposition. *SIAM J. Matrix Anal. Appl*, 16(3):793–810, 1995.

[40] G. Guennebaud, L. Barthe, and M. Paulin. Deferred Splatting. *Computer Graphics Forum*, 23(3):653–660, 2004.

[41] G. Guennebaud and M. Paulin. Efficient screen space approach for hardware accelerated surfel rendering. In T. Ertl, B. Girod, G. Greiner, H. Niemann, H.-P. Seidel, E. Steinbach, and R. Westermann, editors, *Proceedings of the Conference on Vision, Modeling and Visualization 2003 (VMV-03)*, pages 485–494, Berlin, November 19–21 2003. IEEE Signal Processing Society.

[42] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *Proceedings of ACM SIGGRAPH*, pages 325–334, 1999.

[43] P. Hanrahan. Stream programming environments. In *GP2: 2004 ACM Workshop on General-Purpose Computing on Graphics Processors*, pages A–4, 2004.

[44] P. Hanrahan and P. Haeberli. Direct WYSIWYG painting and texturing on 3D shapes. In *Proceedings of ACM SIGGRAPH*, pages 215–223, 1990.

[45] H. Hauser. Generalizing focus+context visualization. In G. M. Nielson G.-P. Bonneau, T. Ertl, editor, *Dagstuhl Conference on Scientific Visualization: Extracting Information and Knowledge from Scientific Datasets*, pages 305 – 327, 2003.

[46] J. M. Henderson. Human gaze control during real-world scene perception. *Trends in Cognitive Science*, 11(7):498–504, 2003.

[47] J. M. Henderson and A. Hollingworth. *Eye movements during scene viewing: An overview. In Eye Guidance in Reading and Scene Perception*. Elsevier Science Ltd., 1998.

[48] M. Hisada, A. G. Belyaev, and T. L. Kunii. A skeleton-based approach for detection of perceptually salient features on polygonal surfaces. *Computer Graphics Forum*, 21(4):689–700, 2002.

[49] S. Howlett and C. O'Sullivan. Predicting and evaluating saliency for simplified polygonal models. *ACM Transactions on Applied Perception*, 2(3):286 – 308, 2005.

[50] V. Interrante, H. Fuchs, and S. Pizer. Conveying the 3D shape of smoothly curving transparent surfaces via texture. *IEEE Trans. on Visualization and Computer Graphics*, 3(1):98–117, April 1997.

[51] V. Interrante and C. Grosch. Visualizing 3D flow. *IEEE Computer Graphics & Applications*, 18(4):49–53, July – August 1998.

[52] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. on Pattern Analysis and Machine intelligence*, 20(11):1254–1259, 1998.

[53] C. Johnson, R. Moorhead, T. Munzner, H. Pfister, P. Rheingans, and T. S. Yoo. NHI-NSF visualization research challenges report. Technical report, Mitsubishi Electric Research Laboratories, 2006. Computing in Science and Engineering,.

[54] D. S. Johnson, S. Krishnan, J. Chhugani, S. Kumar, and S. Venkatasubramanian. Compressing large boolean matrices using reordering techniques. In *International Conference on Very Large Data Bases (VLDB)*, pages 13–23, 2004.

[55] T. Jones, F. Durand, and M.Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics*, 22(3):943–949, 2003.

[56] A. Joshi and P. Rheingans. Illustration-inspired techniques for visualizing time-

varying data. In *IEEE Visualization*, pages 679–686, 2005.

[57] A. Kalaiah and A. Varshney. Modeling and rendering points with local geometry. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):30–42, 2003.

[58] A. Kaufman. *Volume visualization*. IEEE Computer Society Press tutorial. IEEE Society Press, Los Alamitos, CA, 1991.

[59] Y. Kho and M. Garland. User-guided simplification. In *Proceedings of Symposium on Interactive 3D Graphics*, pages 123–126, 2003.

[60] S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2004.

[61] B. Kim and J. Rossignac. Geofilter: Geometric selection of mesh filter parameters. *Computer Graphics Forum (Eurographics 2005)*, 24(3):295 – 302, 2005.

[62] G. Kindlmann and J. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *IEEE Symposium on Volume Rendering Proceedings*, pages 79–86, 1998.

[63] G. L. Kindlmann, R. T. Whitaker, T. Tasdizen, and T. Möller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *IEEE Visualization*, pages 513–520, 2003.

[64] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Trans. on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[65] J. Kniss, S. Premoze, M. Ikits, A. E. Lefohn, C. Hansen, and E. Praun. Gaussian transfer functions for multi-field volume visualization. In *IEEE Visualization*,

pages 497–504, 2003.

[66] C. Koch and S. Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. *Human Neurobiology*, 4:219–227, 1985.

[67] H. Landis. Production-ready global illumination. *Course 16 notes, SIGGRAPH 2002*, 2002.

[68] C. H. Lee, X. Hao, and A. Varshney. Light collages: Lighting design for effective visualization. In *Proceedings of IEEE Visualization*, pages 281–288, 2004.

[69] C. H. Lee, A. Varshney, and D. Jacobs. Mesh saliency. *ACM Trans. on Graphics (Procs. ACM SIGGRAPH)*, 24(3):659 – 666, 2005.

[70] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[71] M. Levoy and T. Whitted. The use of points as a display primitive. In *Technical Report 85-022, Computer Science Department, UNC, Chapel Hill*, 1985.

[72] G. Li and B. Watson. Semi-automatic simplification. In *Proceedings of Symposium on Interactive 3D Graphics*, pages 43–48, 2001.

[73] D. Lischinski and A. Rappoport. Image-based rendering for non-diffuse synthetic scenes. In *Rendering Techniques '98*, Eurographics, pages 301–314, 1998.

[74] A. Lu, R. Maciejewski, and D. Ebert. Volume composition using eye tracking data. In *Eurographics/IEEE VGTC Symposium on Visualization*, pages 147–154, 2006.

[75] A. Lu, C. Morris, D. S. Ebert, P. Rheingans, and C. D. Hansen. Non-photorealistic volume rendering using stippling techniques. In *IEEE Visualization*, pages 211–218, 2002.

[76] D. Luebke and B. Hallen. Perceptually-driven simplification for interactive ren-

dering. In *Proceedings of the Eurographics Workshop on Rendering Techniques*, pages 223–234, 2001.

[77] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan Kaufman, 2003.

[78] E. B. Lum and K. Ma. Lighting transfer functions using gradient aligned sampling. In *IEEE Visualization*, pages 289–296, 2004.

[79] R. Machiraju, J. E. Fowler, D. Thompson, W. Schroeder, and B. Soni. Evita: Efficient visualization and interrogation of tera-scale data. *Data Mining for Scientific and Eng. Applications*, pages 257–279, 2001.

[80] S. Manna, K. H. Ruddock, and D. S. Wooding. Automatic control of saccadic eye movements made in visual inspection of briefly presented 2-d images. *Spatial Vision*, 9(3):363, 1998.

[81] R. Mantiuk, K. Myszkowski, and S. Pattanaik. Attention guided MPEG compression for computer animations. In *Procs. 19th Spring Conference on Computer Graphics*, pages 239–244, 2003.

[82] S. McCloud. *Understanding Comics – The Invisible Art*. Harper Perennial, 1994.

[83] M. Meyer, M. Desbrun, P. Schr, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and Mathematics III*, pages 35–57, 2003.

[84] R. Milanese, H. Wechsler, S. Gil, J. Bost, and T. Pun. Integration of bottom-up and top-down cues for visual attention using non-linear relaxation. In *IEEE Computer Vision and Pattern Recognition*, pages 781–785, 1994.

[85] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for

detail-preserving mesh editing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 24(3):1142–1147, 2005.

[86] A. Ni, K. Jeong, S. Lee, and L. Markosian. Multi-scale line drawings from 3D meshes. In *Proceedings of Symposium on Interactive 3D Graphics*, pages 133–137, 2006.

[87] M. Nienhaus and J. Dollner. Depicting dynamics using principles of visual art and narrations. *IEEE Computer Graphics and Applications*, 25(3):40–51, 2005.

[88] A. Oliva, A. Torralba, M. S. Castelhano, and J. M. Henderson. Top-down control of visual attention in object detection. In *International Conference on Image Processing*, pages 253–256, 2003.

[89] R. Pajarola. Efficient level-of-details for point based rendering. In *Proceedings IASTED Computer Graphics and Imaging Conference*, pages 141–146, 2003.

[90] S. Palmer. *Vision Science: Photons to Phenomenology*. MIT Press, Cambridge, MA, 1999.

[91] D. Parkhurst, K. Law, and E. Niebur. Modeling the role of salience in the allocation of overt visual attention. *Vision Research*, 42(1):107–123, 2002.

[92] D. Parkhurst and E. Niebur. Texture contrast attracts overt visual attention in natural scenes. *European Journal of Neuroscience*, 3(19):783 – 789, 2004.

[93] M. Pauly, M. H. Gross, and L. Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization*, pages 163–170, 2002.

[94] R. Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003.

[95] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. S. Avila, K. Mar-

tin, R. Machiraju, and J. Lee. The transfer function bake-off. *IEEE Comput. Graph. Appl.*, 21(3):16–22, 2001.

[96] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of SIGGRAPH 2000*, pages 335–342, 2000.

[97] G. Pingali, A. Opalach, Y. Jean, and I. Carlbom. Visualization of sports using motion trajectories: Providing insights into performance, style, and strategy. In *Proceedings Visualization 2001*, pages 75–82, 2001.

[98] M. Posner. Orienting of attention. *Quarterly Jnl. of Exper. Psych.*, 32:3 – 25, 1980.

[99] C. M. Privitera and L. W. Stark. Algorithms for defining visual regions-of-interest: Comparison with eye fixations. *IEEE Trans. on Pattern Analysis and Machine Intellignece*, 22(9):970–982, 2000.

[100] G. N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. Stereochemistry of polypeptide chain configurations. *Journal of Mol. Biol.*, 7:95–99, 1963.

[101] B. Srinivasa Reddy and B. N. Chatterji. An FFT-based technique for translation, rotation, and scale-invariant image registration. *IEEE Transactions on Image Processing*, 5(8):1266–1271, 1996.

[102] L. Ren, H. Pfister, and M. Zwicker. Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. In *Eurographics'02*, pages 461–470, September 2002.

[103] A. Reshetov, A. Soupikov, and J. Hurley. Multi-level ray tracing algorithm. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005)*, 24(3):1176–1185, 2005.

[104] P. Rheingans and D. Ebert. Volume illustration: Nonphotorealistic rendering of

volume models. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):253–264, 2001.

[105] R. Rosenholtz. A simple saliency model predicts a number of motion popout phenomena. *Vision Research*, 39(19):3157–3163, 1999.

[106] S. Rusinkiewicz, M. Burns, and D. DeCarlo. Exaggerated shading for depicting shape and detail. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 25(3):1199–1205, 2006.

[107] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of SIGGRAPH 2000*, pages 343–352, 2000.

[108] A. Santella and D. DeCarlo. Visual interest and NPR: an evaluation and manifesto. In *Proceedings of NPAR*, pages 71–150, 2004.

[109] J. Shade, S. Gortler, L. He, and R. Szeliski. Layered depth images. In *Proceedings of SIGGRAPH 98*, pages 231–242. ACM Press / ACM SIGGRAPH, August 1998.

[110] Q. Shi and J. JaJa. Isosurface extraction and spatial filtering using persistent octree (POT). *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1283–1290, 2006.

[111] P. Shilane and T. A. Funkhouser. Distinctive regions of 3D surfaces. *ACM Trans. Graph*, 26(2):7, 2007.

[112] E. P. Simoncelli and W. T. Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *International Conference on Image Processing*, pages III: 444–447, 1995.

[113] D. Stampe. Heuristic filtering and reliable calibration methods for video-based pupil tracking systems. *Behavior Research Methods, Instruments and Computers*,

25:137 – 142, 1993.

[114] S. L. Su, F. Durand, and M. Agrawala. De-emphasis of distracting image regions using texture power maps. In *Texture 2005: Procs. 4th IEEE International Workshop on Texture Analysis and Synthesis in conjunction with ICCV'05*, pages 119–124, October 2005.

[115] S. Sukharev and A. Anishkin. Private communication, 2008.

[116] V. Surazhsky and C. Gotsman. Explicit surface remeshing. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 20–30, 2003.

[117] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 24(3):553–560, 2005.

[118] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings of IEEE International Conference on Computer Vision*, pages 902–907, 1995.

[119] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of ACM SIGGRAPH*, pages 351–358, 1995.

[120] A. Torralba. Modeling global scene factors in attention. *Journal of Optical Society of America A*, 20(7):1407–1418, 2003.

[121] S. M. F. Treavett and M. Chen. Pen-and-ink rendering in volume visualisation. In *Proceedings Visualization 2000*, pages 203–210, 2000.

[122] J.K. Tsotsos, S.M. Culhane, W.Y.K. Wai, Y.H. Lai, N. Davis, and F. Nuflo. Modeling visual-attention via selective tuning. *Artificial Intelligence*, 78(1-2):507–545,

1995.

[123] G. Turk and D. Banks. Image-guided streamline placement. In *Proceedings of SIGGRAPH 1996*, pages 453–459, 1996.

[124] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven feature enhancement in volume visualization. *IEEE Trans. on Visualization and Computer Graphics*, 11(4):408–418, 2005.

[125] J. Vorsatz, C. Rössl, L. Kobbelt, and H.-P. Seidel. Feature sensitive remeshing. *Computer Graphics Forum*, 20(3):393–401, 2001.

[126] Q. Wang and J. JaJa. Interactive high-resolution isosurface ray casting on multicore processors. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):603–614, 2008.

[127] K. Watanabe and A. G. Belyaev. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum (Eurographics 2001)*, 20(3):385–392, 2001.

[128] T. Welsh and K. Mueller. A frequency-sensitive point hierarchy for images and volumes. In *IEEE Visualization'03*, pages 425–432, October 2003.

[129] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. In *Advances in Computational Mathematics*, volume 4, pages 389–396, 1995.

[130] H. Winnemöeller, S. C. Olsen, and B. Gooch. Real-time video abstraction. *ACM Trans. Graph.*, 25(3):1221–1226, 2006.

[131] J. C. Woolley, D. Luebke, and B. Watson. Interruptible rendering. In *SIGGRAPH'02 Technical Sketch*, page 205. ACM Press / ACM SIGGRAPH, 2002.

[132] J. Wu and L. Kobbelt. Optimized sub-sampling of point sets for surface splatting. volume 23, pages 643–652, 2004.

[133] H. Yamauchi, W. Saleem, S. Yoshizawa, Z. Karni, A. Belyaev, and H.-P. Seidel. Towards stable and salient multi-view representation of 3D shapes. In *Proceedings of Shape Modeling International*, pages 265–270, 2006.

[134] Y.-L. Yang, Y.-K. Lai, S.-M. Hu, and H. Pottmann. Robust principal curvatures on multiple scales. In *Eurographics Symposium on Geometry Processing*, pages 223–226, 2006.

[135] H. Yee, S. Pattanaik, and D. P. Greenberg. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Transactions on Graphics*, 20(1):39–65, 2001.

[136] H. Zhang and E. Fiume. Mesh smoothing with shape or feature preservation. In *Advances in Modeling, Animation, and Rendering*, pages 167 – 182, 2002.

[137] S. Zhukov, A. Inoes, and G. Kronin. An ambient light illumination model. *Proceedings of Eurographics Rendering Workshop*, pages 45–56, 1998.

[138] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of the ACM Conference on Computer Graphics*, pages 189–192, 1996.

[139] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *Proceedings of SIGGRAPH 2001*, pages 371–378, 2001.