# LIMITED RESOURCE VISUALIZATION WITH REGION-OF-INTEREST

YU HANG

NATIONAL UNIVERSITY OF SINGAPORE

2006

**Name: Yu Hang**

**Degree: Doctor of Philosophy**

**Dept: Department of Computer Science**

**Thesis Title: Limited resource visualization with region-of-interest**

## Abstract

This thesis studies some issues on applying region-of-interest in visualization. In visualization, a critical consideration is on how to handle very large data-set with limited resources, specifically computational resources and display window size. Region-of-interest (ROI) technique can be employed as a potential solution to serve the following two purposes: 1) It allocates more computational resources to the interesting region. 2) It assists the viewer by filtering out less interesting information. In this thesis, we study the above issues in the context of two applications: remote volume visualization with limited computational resources at the client side, and vector map visualization in small display window. For the first application, a technical issue is on how to apply ROI on volume visualization efficiently. This is important in scenarios where the viewer has access to low computational resources. Another issue is on how to apply ROI effectively. We give several methods to adjust the transfer function to highlight objects in the ROI. For the second application, consideration should be given on how to present the local and global geographic information simultaneously in the limited display window. We give a map generalization method that first adopts fisheye view to exaggerate information in ROI followed by a line smoothing process to eliminate the clutter caused by the distortion. The smoothing process is essentially an iteration of localized smoothing processes that maintain the topological consistency.

**Keywords:** Visualization, Region-of-interest, Wavelet foveation, Fisheye view

# LIMITED RESOURCE VISUALIZATION WITH REGION-OF-INTEREST

**YU HANG**

*(M.E., Shanghai JiaoTong University, China)*

*(B.E., Shanghai JiaoTong University, China)*

**A THESIS SUBMITTED**

**FOR THE DEGREE OF DOCTOR OF PHILOSOPHY**

**DEPARTMENT OF COMPUTER SCIENCE**

**NATIONAL UNIVERSITY OF SINGAPORE**

**2006**

LIMITED RESOURCE VISUALIZATION WITH REGION-OF-INTEREST

YU HANG

2006

# Acknowledgements

I would like to deliver my deep appreciation to my adviser Dr. Chang Ee-Chien. With his encouragement and patience, I could get across the difficult times for completing this thesis. His insight and knowledge help me much to build my research capabilities.

I would like to thank my thesis committee members for their support and valuable comments.

Finally, I would like to thank my family with their loving support.

# Contents

# Summary

Region-of-interest (ROI) technique can be employed in visualization to serve two purposes: 1) It allocates more computational resources to the interesting region. 2) It assists the viewer by filtering out less interesting information. This technique offers a compromise between efficiency and accuracy, thus improving the responsiveness during real-time visualization or decision making process. Typically, ROI technique divides the investigated data into two regions: an emphasized region of high-interest, and the remaining suppressed region. It is not necessary to have only two regions. To achieve a smooth transition from high to low level of interest, one could incorporate foveation, or a fisheye view transformation. In this thesis, we study ROI with foveation or fisheye view, in the context of two applications: remote volume visualization with limited computational resources at the client side, and vector map visualization in small display window.

In the first part of the thesis, we focus on foveated volume. A technical issue is on how to render a foveated volume efficiently. This is important especially in the remote visualization setting where a low computing device is connected to a server storing the volume data. We give an algorithm that renders a foveated volume directly in the wavelet domain. The number of wavelet coefficients representing the foveated volume is significantly smaller than the number of voxels. Another issue is on how to visualize a foveated volume effectively. We give several methods to adjust the transfer function to highlight objects in the ROI.

In the second part, we study visualization of vector-based map in a small window. Due to the limited size of display window, consideration should be given to the presen-

tation of the geographic information that contains both the focus and the context of the surrounding region. We give a method that adopts fisheye view transformation to magnify information in ROI, and a smoothing process to eliminate the clutter caused by the distortion. The smoothing process is essentially an iteration of localized smoothing processes that meet the topological constraints.

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background

The term "visualization" has been defined differently in various domains of science. According to the 1989 Oxford English Dictionary, visualization is defined as "the formation of mental visual images, the act or process of interpreting in visual terms or of putting into visual form."

The strength of visualization lies in the fact that huge amounts of intricate data can be interpreted as refined information for humans. As described by a common saying: "An image is worth than a thousand words", visual representation of data is more meaningful to human than other formats e.g. text or audio. Visualization helps to equip people with the ability to see the "unseen" [67], thus providing new insights into information. Visualization can be classified into three categories: scientific, information and data visualization. Scientific visualization studies the visual representation techniques of scientific data from physical reality or process. In contrast to scientific visualization, information visualization processes abstract data which are usually not mapped into physical world. Data visualization is a more general term that handles data beyond science and also includes data analysis techniques. The power of visualization has made it widely applied in many domain of applications as follows.

- **Medical imaging and visualization.** For applications in medical field, visualization is utilized as the tool to investigate internal organs of subjects. Anatomical information is acquired by various imaging technologies such as CT (Computer Tomography ), MRI (Magnetic Resonance Imaging) or PET (Positron Emission Tomography). To present the information, there are two conventional visualization techniques: volume rendering and iso-surface extracting. The distinction between these two is that the former one can process the whole data, both inner structures and surfaces. Additionally, nonphotorealistic rendering techniques have been studied in medical visualization. Based on pen-and-ink illustration, the methods aim to enhance features (e.g. silhouette, boundary) of medical data.

- **Geographic visualization.** Geographic visualization models ground features including natural features (e.g. mountains, valleys and rivers, etc.) and man-made features (e.g. buildings, roads and rails, etc.) with geometric symbols. Generally, geographic information is represented by two approaches: layer-based and feature-based. Layer-based approach models spatial data by a set of layers containing independent information, such as water-, mountain-, transportation- system etc. The layers can be combined to form a map with different themes. Feature-based approach is also called entity-based. A feature is used to describe spatial attribute of geographic entities, such as river, road, boundary, etc. For visualization, each feature is explicitly represented by their corresponding geometric symbols, e.g. point, line, polygon, etc.

- **Computational fluid dynamics and visualization.** In the field of computational fluid dynamics, visualization is the process to reveal dynamic characteristics of flows such as liquids or gases. The visualization approaches can be classified as direct-, texture-, geometric-, feature-based approaches. Direct-based approach is quite straightforward to depict flows by drawing techniques, such as arrow plots. Texture-based approach attempts to give a dense representation of flows by mapping textures in the vector field. Geometric-based approach is applied after the

integration of flow data. Geometric objects are used to render the integrated flows in order to study their long-term behavior. Feature-based approach is performed before visualization to extract features from flow data. Efficient visualization can be achieved based on the extracted flow features.

- **Time-dependent visualization.** Visualization of time-dependent data is applied to analyze non-static process in scientific applications. Visualizing by animation is a simple approach which gives snapshots of time-varying data at sequential time step. This approach may not handle very large data-sets. Feature tracking is an efficient approach to extract and track region-of-interest during the process of time.

- **Abstract information visualization.** Visualization of abstract information mainly deals with developing visual representation of unscientific data, for e.g. file documents, relationships in databases. Conventionally, such data is displayed by various graph drawing approaches, such as plots, charts or histograms. However these techniques are unable to handle large and high-dimensional data. Some sophisticated techniques have been proposed to cater the limitations [46]. According to the display mode, they are classified into five classes: standard 2D/3D displays which are conventional approaches; geometric transformation displays apply geometric projection on the visualized data; icon-based displays visualize data values as feature icons; dense pixel displays treat data values in each dimension as color pixels which are clustered for visualization; stacked displays particularly handle data which are represented in a hierarchical way.

- **Virtual reality and visualization.** Virtual reality aims to provide human with a computer generated experience of realistic or imaginary world. Through a set of combined computer technologies, a virtual environment is generated to interact with human. Besides the accessorial devices, visualization is an important technology that presents virtual reality to human. Currently, the general visual-

izing approaches applied in virtual reality system are computer-assisted design, computer graphics and animations.

- **Remote visualization.** Due to the popularity of the Internet and mobile services, there is a growing interest and demand of visualizing data stored in a remote server. It is applied when the data are difficult to process in local resources or collaborations among a group are required. Generally, there are two strategies of remote visualization: render-local which transmits raw data to viewers to process and visualize; render-remote which only transmits processed results to viewers. In remote visualization, real-time data transfer is a challenging issue. To meet this requirement, one possible solution is to use progressive transmission and refinement. Besides the transmission bandwidth, the low computing power of the client is also a concern for processing large volume data. For example, to process a data-set with $512 \times 512 \times 512$ voxels, is infeasible for most general purpose desktop PCs.

## 1.2 Research directions

### 1.2.1 Research scope

This thesis intends to study selected issues in visualization with ROI where the viewer has limited resources. The resources can be in the form of computing power, or even the size of the display window. The role of ROI is to allocate more resources to the interesting region.

In remote volume visualization, a promising technique streams the volume starting with regions providing higher level of interests. This results in a foveated volume which has highest resolution at the point of focus. In order to display the up-to-date data, a straightforward method would continuously reconstruct the volume from the received raw data, and then render it. This is computational intensive and not suitable for a client with limited computing resources. Hence, a goal of the first part of this thesis in

Chapter 2 is to design an algorithm that can render a foveated volume directly from the received wavelet coefficients. From another perspective, it is not clear on how to display a volume with multiple levels of resolution. Should the voxels occluding the ROI be rendered in lower details, totally removed, or treated to be translucent? Another goal of the work is to give a few ways to visualize a foveated volume.

In the second part of the thesis, we treat display window size as a resource and study how to exploit the small window using ROI. This is particularly relevant in the application of map browsing with mobile device which typically has small window. A natural solution is to apply a fisheye transformation to magnify information in ROI and suppress the rest. However, the distortion caused by the operation may result in information clutter. Hence, our goal is to provide a map generalization method that can present the focus plus context map presentation without information clutter in a small display.

### 1.2.2 Main contributions

**Remote volume visualization**

- In this work, we adopted the notion of wavelet foveation [16] to obtain a compact wavelet-based representation of a multiple levels-of-detail volume. We gave an efficient algorithm that renders a foveated volume directly in the wavelet domain. We exploited the arrangement of the relevant wavelet coefficients to achieve fast rendering. The running time only depends on the number of relevant wavelet coefficients. Specifically, the running time is $O(n^2 + m)$, where $n$ is the width of the volume data, and $m$ is the number of relevant wavelet coefficients. This is an improvement compared to the straightforward rendering in the spatial domain that requires $O(n^3)$ running time.

- We gave several methods to adjust the transfer function to highlight objects in the fovea. By this way, the viewer's attention is directed to the fovea. This is achieved

by multiplying the original opacity with a space-variant weighting function. Hence the opacity of a voxel depends both on its location and intensity.

- A side-result in this thesis is a method that rotates a foveated image/volume efficiently in the wavelet domain. This method is an extension of a component in the foveated volume rendering that handles non-orthogonal viewing direction. An efficient rotation directly in the wavelet domain could be useful in other applications, for example computer vision with foveated images.

**Vector map visualization**

We proposed a map generalization method with the following three steps: 1) According to current navigation task, non-related map objects are filtered and excluded from the visualization process. 2) Fisheye view transformation is applied to magnify information in ROI while suppressing the surrounding region. 3) The lines are smoothed to eliminate information clutter caused by the geometric distortion in the second step.

In step 3, we treated the smoothing as an optimization problem which minimizes the curvature and distortion, while preserving the area of individual subregion. We gave a heuristic method to find a solution. Our heuristic method iteratively solves a sub-problem: Given two curves which intersect at most once in a circular domain, find two Bézier curves such that the partitioned areas are preserved.

## 1.3 Thesis organization

The thesis is organized as follows. Chapter 2 addresses ROI techniques used in remote volume visualization. It first introduces the background of volume visualization and related work. Next it describes the proposed fast volume rendering algorithm based on foveation. Additionally, it discusses two ways for foveated volume visualization. Finally, it gives some potential applications of the algorithm in remote visualization.

Chapter 3 gives the side-result of our proposed method in Chapter 2. It studies the rotation of a foveated image/volume in the wavelet domain. It is applied in Chapter 2

to handle rendering with non-orthogonal viewing directions.

Chapter 4 illustrates ROI techniques used in geographic vector map visualization on small display window. It first gives literature reviews. Following this, it presents the algorithm and experimental results.

Chapter 5 gives the conclusions of the thesis.

# Chapter 2

# Volume visualization using region-of-interest

## 2.1 Introduction and related work

Volume visualization is an efficient technique to analyze and reveal important interior information in many scientific applications. For example, in medicine, medical volume data obtained by CT (computed tomography) and MRI (magnetic resonance imaging) scanners act as a valid reference to examine the inner structures of patients' organs [32]. In geo-science, volume visualization is used as a method to analyze information retrieved by seismic instruments to investigate the composition of the earth [30]. Volume visualization also finds its application in computational fluid dynamics to simulate fluid movement in 3D space [23].

The complete process of volume visualization may consist of many steps [44]. Fundamentally, there are four steps commonly used.

The first step is data acquisition. This step involves activities to collect data through either measurement devices such as CT and MRI scanners or computer simulation. When the raw data are generated, the next step is to transform them before any visualization algorithm can apply on them. The objective of this step is to put the data

into some appropriate format for easy manipulation. The following step is to map the processed data onto geometric or display primitives. This step may vary distinctly by different algorithms. The final step is to store, manipulate or display the primitives.

### 2.1.1 Volume visualization techniques

Generally, volume visualization techniques are classified into two categories: surface rendering (SF) and direct volume rendering (DVR).

Surface rendering method is also known as iso-surface extracting. It generates the constant-value contour surfaces in volume data by extracting data values with geometric primitives, such as polygon meshes or surface patches. In order to visualize the whole data, animation is required on the sequence of iso-surfaces given different thresholds. Existing methods of surface rendering include contour connecting [24], opaque cubes [37], marching cubes [57], dividing cubes [17] and marching tetrahedral [93]. Typically, SF methods are faster than DVR methods as the former only traverse once over the volume data to create surfaces. However a restriction of the methods is that they are only effective when the iso-surfaces of underlying data are smooth and simple. They may not handle data with irregular structure, such as liquid or gas.

In contrast, DVR methods directly map volume data onto display primitives without the assistance of any geometric structure. By these methods, all the information contained in the data is rendered thus a more comprehensive representation is obtained than SF methods. Obviously, it is the reason that they are slower than SF methods. Common DVR methods include ray casting [54], splatting [113], shear-warp factorization [49], etc.

#### Optical models

We now give a detailed description of direct volume rendering, since we adopt it in our rendering algorithm. In direct volume rendering, a model is required to formulate the process of light absorption and emittance through the volume data. As a complete formulation of the interaction between light and modeled volume particles is

non-practical, many simplified models are designed to achieve a good approximation. One of the first optical models was developed by Blinn [9]. Blinn's model was designed to study the optical properties of the clouds of ice particles that build up the rings of the Saturn. In his method, the interaction (reflection and transmission) between light and the particles was modeled by single reflection approximation. Alternative models were given by several researchers [42, 66, 84, 22]. Max gave a detailed review of the different optical models [65].

The optical model adopted in this thesis is Max's emission-absorption optical model [65]. Under this model, the light traversing a volume density is both emitted and absorbed. The approximation for the volume rendering integral equation is given as follow [65]:

$$I(t_1, t_2) = \int_{t_1}^{t_2} V(t) e^{-\int_{t_1}^{t} \alpha(s) ds} dt \tag{2.1}$$

where $I$ is the resulting intensity for the light along viewing rays to the viewer, $t_1$ and $t_2$ are the start and end points on the viewing ray, $V(t)$ is the intensity value at location $t$, and $\alpha(s)$ is the opacity at $s$.

In the discrete case, each sample in the volume is called a voxel. E.q. 2.1 can be reduced to a finite sum over the accumulated opacity with the assumption that the intensity function and opacity function for a certain segment $i$ are constants as $v_i$ and $\alpha_i$. This gives:

$$I = \sum_{k=1}^{n} v_k \alpha_k \prod_{i=0}^{k-1} (1 - \alpha_i) \tag{2.2}$$

**Direct Volume Rendering Algorithms**

Volume rendering involves the process to generate the projection of 3D volume data-set and display the rendering results on a 2D image plane for viewers. The process

includes the following three steps:

- Mapping optical properties (color, opacity) concerning the interaction with the light to the volume data element. The data element in volume is named as voxel. Each voxel is assigned the color and opacity based on its intensity, gradient magnitude or gradient direction, etc. Usually such process is realized by designing elaborative transfer functions. Finding an appropriate transfer function is quite tricky as it may depend on a large amount of experimental trials.

- Integrating the overall contribution from each voxel along the light that casts into one pixel in the 2D image plane. When the light traverses the volume, it is absorbed and emitted by the voxels that it steps cross. The accumulated rendering over the volume is the integral of both the color and opacity composition along the light. In most volume rendering algorithms, the voxels are distributed in the structure of a 3D grid space. Re-sampling is required when the light does not pass exactly through the grid node. The common re-sampling approaches include nearest-neighbor, tri-linear interpolation and tri-cubic interpolation.

- Projecting the rendering results onto 2D image plane according to different viewing directions. There are two kinds of projection modes: parallel projection and perspective projection. By parallel projection, all the virtual rays that simulate the light casting into the volume are parallel to each other. In the mode of perspective projection, the virtual rays are casted from a point which is at a finite distance from the volume.

The existing direct volume rendering algorithms can be classified into the following four categories: image-order based, object-order based, the hybrid of image- and object-based and domain based. This classification is based on the data traversing order and how the data are processed during volume rendering.

Image-order based technique is also called backward mapping. In this method, the virtual rays are casted from the viewer through pixels in the image plane across into

the volume. Integration of the color and opacity is performed when the rays intersect with the volume voxels. Ray-casting is a representative algorithm in this class [53, 103, 55, 54, 42]. Besides the common color and opacity blending, there are several other rendering effects. It may be the X-ray rendering that simply sums up the data values along the rays or maximum intensity projection that only selects the maximum data value for each ray. Ray-casting is rather straight-forward and can produce high quality images, however it is quite time-consuming when the number of volume voxels is large. To improve the performance of ray casting, some approaches were proposed such as skipping regions which are not contributing to the rendering (e.g. transparent regions) [114], early ray termination that terminates the ray when the accumulated opacity has reached a given thresholded value [54].

Object-order based technique is forward mapping in contrast to image-order technique. The volume data samples are projected onto the image plane. In such a way, there is only one-time computation for all the voxels contributing to a set of pixels consisting of a region in the image plane. A good example of object-order based techniques is splatting [113, 112]. Splatting can achieve faster rendering speed than ray-casting as it only processes relevant voxels contributing to the image. As a price, it may produce lower quality rendering. Optimization strategies for splatting were developed by utilizing object- and image- space coherence that exclude non-interesting regions or regions occluded by others [41, 71, 51].

By combining the advantages of the two approaches mentioned above, hybrid technique was proposed to achieve good rendering quality as well as fast rendering speed. Shear-warp factorization [49] is one of such methods and among the fastest software-based methods so far. The main idea of this method is that volume data are transformed into a sheared object space with all the rays paralleling to the principal coordinate axis. With this transformation, the advantage is that it avoids the expensive tri-linear interpolation by bi-linear interpolation. The projection through the sheared volume is distorted and can be corrected by a 2D warping for the final results. The drawback of

this method is that it may require additional memory storage to create volume stacks for the three viewing coordinate axises.

For domain-based method, the spatial volume data are represented by some alternative domains, such as frequency, compression and wavelet domain, and rendered directly from these domains. The motivation for switching to render in other domains is to achieve faster rendering speed and lower computational complexity.

Frequency domain-based rendering was first proposed by Dunne et al. [21] and further extended by Malzbender [63] and Totsuka et al. [101]. This approach is based on the Fourier projection-slice theorem [43] that the rendered image perpendicular to the viewing direction can be generated by extracting a 2D slice from the 3D Fourier-transformed data and transforming back to the spatial domain. The advantage is that it achieves much low computational complexity as it avoids the conventional rendering integral along the viewing direction. However there is a drawback for this approach: it is not possible to change the transfer functions interactively after the volume data are transformed into the frequency domain.

Compression domain-based rendering provides volume rendering in the compressed domain and there is no need to decompress all the data for the rendering. Thus the storage and computation requirement are reduced. An example of this class is the work by Ning and Hesselink [76]. Their work used vector quantization to give a lossly compressed representation of volume data. The rendering is performed directly based on a relatively small codebook. Yeo and Liu [115] gave a method based on discrete cosine transform (DCT). The volume data are divided into blocks and compressed by 3D DCT. In the process of rendering, only relevant blocks are decompressed. A more recent work was presented by Fout et al. [29] that performed rendering from compressed volume data by deferred filtering. The compressed data were first decompressed into slices and filtered for rendering.

Wavelet analysis is an important technique widely used in image compression and signal processing for its time-frequency localization feature. It is also applied in volume

rendering for the purpose of compression, progressive transmission and multi-resolution rendering, etc. The idea of wavelet-based volume rendering was first introduced by Muraki [74] who applied 3D wavelet transformation to approximate volume data. Westermann presented volume rendering based on wavelet compression [111]. Gross et al. gave a method of wavelet splatting that performed rendering directly on wavelet coefficients of volume data [33].

In order to handle non-orthogonal viewing directions, shear-warp factorization [49] is an efficient approach. Under this transformation, the volume is sheared such that each slice of it is perpendicular to the viewing direction. Thus it is identical to apply the orthogonal projection on the sheared volume.

However this approach can not be directly applied in wavelet-based volume rendering. This is because the factorization can not be performed in the wavelet domain due to the inherent disadvantages of discrete wavelet transform. One of the disadvantages is "shift sensitive", i.e. a small shift in 1-dimensional signal generates unpredictable changes in its discrete wavelet transform coefficients. This is caused by the down sampling operations in the transform. To overcome the "shift sensitive", undecimated DWT was devised by Mallat [62] that removed the down sampling operations. But this solution was relatively expensive as it introduced high transform redundancy.

The shift insensitive complex wavelets [27] could be employed to handle the translation. However, it is not clear how they can be applied to handle slightly more complicated image operations like rotation and shearing. Also note that in general, even if the original data have many zero coefficients, the operated images could have few or none zero coefficients. Therefore, it is impossible to have both fast and exact algorithm.

### 2.1.2 ROI techniques in volume rendering

There are many work in the direction of region-of-interest based visualization. Furnas [31] introduced the concept of fisheye view by presenting information with a magnifying glass effect. As a result, the important information is displayed in much detail while

the context is demagnified further away. Following Furnas's work, several strategies have been developed [61, 82, 86]. With these techniques, a fast rendering rate can be achieved by allocating different priority among the spatial domain of the data-set.

Several research work have studied on the multi-levels ROI rendering of volume data-sets. Levoy et al. [56] gave a real-time volume rendering system that rendered volumes in two different levels of resolution. These two rendered images were then blended to obtain the final rendered image. Piccand et al. [81] described a method to perform X-ray projection in the wavelet domain, such that the ROI was projected in full resolution, while other voxels were projected in reduced resolution. Along a viewing ray that entered the ROI, voxels lying before or after the ROI were omitted in the projection. The main technique employed by Piccand et al. was wavelet splatting [50], which pre-computed a 2D projected *footprint* for each sub-band.

From another perspective, an interactive visualization session can be more effective if the objects in the ROI are highlighted and information outside the ROI is filtered or reduced. Hence, even if the whole data-set is available, or there are sufficient computing resources, applying ROI in visualization can still be useful. This leads to the issue of how to effectively visualize a volume with a point of focus. Zhou et al. [117] proposed to use distance as a factor to adjust objects' opacity. Viola et al. [105] presented a technique that suppressed less important information in volume rendering by cutting away objects occluding the interesting objects.

### 2.1.3 Wavelet-based foveation

This thesis adopts foveation as a variation of ROI techniques for volume data visualization. Foveation is the biological process of human visual system (HVS) to non-uniformly sample the world that the resolution is highest at the fovea but falls off as the distance from the fovea increases [91]. This is due to the space-variant nature of human visual perception. Such mechanism provides an effective way of navigating the visual field by compressing the information without sacrificing visual quality.

In the foveation method, the ROI is indicated as the focus of viewer's gaze point. The implementation of foveation can be achieved by two common approaches: log-polar transform [110, 98] and wavelet foveation [16]. Based on log-polar transformation, foveation is obtained by first applying a log-polar transformation on the visual field, then a convolution in the log-polar space and transformation back into Cartesian space. Wavelet-based foveation gives an alternative way that efficiently approximates the non-uniform sampling process in wavelet domain. As this method is experimentally proved to be fast and accurate [16], it is quite suitable to be applied in the context of remotely visualizing large data-sets.

The special property of foveation has been utilized in many application fields. In some computer vision systems, foveated imaging was applied for active vision [90, 96] in order to provide high resolution on target objects in a wide viewing angle. This achieved much cost reduction and performance improvement in applications like video surveillance or tracking tasks. Foveated visions were widely employed in image transmission [16], video processing [52, 5, 25], flight simulation [28, 99], 3D model visualization [4], volume rendering [56], etc. The main purpose was that by mimicking HVS, a good trade-off was obtained between the visual quality and some performance measures, such as compression rate, transmission cost. From another perspective, foveation can also be viewed as a way to distribute computing resources across space. For example, our early work [116] gave a fast volume rendering algorithm that rendered volumes in multiple levels of resolution.

The mathematical formulation of the "ideal" foveation process has been discussed [16]. We give a brief overview here.

The foveated transformation $f : R^d \rightarrow R$ is controlled by two components: a scaling function $s : R^d \rightarrow R$ and a weight function $w : R^d \rightarrow R_{\geq 0}$.

$$(Tf)(x) = \int_{-\infty}^{\infty} f(t) \frac{1}{w(x)} s\left(\frac{t-x}{w(x)}\right) dt \tag{2.3}$$

The scaling function $s$ controls the compression ratio and is normalized as $\int_{-\infty}^{\infty} s(x)dx = 1$. The weight function $w$ is determined by three parameters as $w(x) = \alpha|x - \gamma| + \beta$. $\alpha$ is called *rate* as it gives the decaying speed of the resolution. $\gamma$ is called *fovea* as it gives the location of the highest resolution. $\beta$ is called *foveal resolution* as it gives the resolution at $\gamma$. Thus $w$ controls the distortion from the fovea to the peripheral. If the two functions $s$ and $w$ are replaced by a kernel function $k(x,t)$ of $T$, E.q. 2.3 can be written as an integral operator

$$(Tf)(x) = \int_{-\infty}^{\infty} f(t)k(x,t)dt \tag{2.4}$$

This foveation operator can be efficiently approximated under wavelet transformation as the transformed kernel is dominated by its diagonal terms. The main idea is that in E.q. 2.4, if the original function $f$ is represented by uniformly sampled $N$ points, the computation of the foveated function $Tf$ can be treated as a matrix multiplication. The running time for the arithmetic operations is $O(N^2)$. When representing the kernel in wavelet domain, most terms in the transformed kernel become small. A sparse matrix is obtained by suppressing these small terms. Operating on the sparse matrix with wavelet transform, the running time is reduced to $O(N)$. Thus a fast algorithm for foveation is possible. The detailed mathematical illustration is given in the work by Chang [15].

### 2.1.4 Potential applications

- **Remote visualization**

  A potential application of our algorithm is in remote volume visualization. A viewer at the client-side indicates the fovea, and the selected coefficients are sent across (alternatively, we can let another viewer at the server-side indicates the fovea). In the client-side, the viewer applies our algorithm to render the obtained

foveated volume. The server continues to send coefficients across, achieving the effect that the fovea rate is increasing. For the viewer, what he/she sees is the rendering result that is getting more and more accurate. Note that if direct rendering method is used here, then the inverse wavelet transformation has to be applied for every new coefficient arriving at the client-side. Our algorithm works efficiently in the wavelet domain and hence overcomes this problem.

- **Time-varying volume data visualization**

  Another application is in the visualization of time-varying volume data. If the time-varying volume data are already represented in a foveated form, it is possible to apply our idea to achieve fast rendering. For example, in a system of video sensor networks, video sensors are spatially distributed to capture and reconstruct a dynamic 3D view of the scene. The coverage of the sensors could be wide and thus impossible to perform a full-resolution real-time scene rendering. As the distribution of the sensors in the 3D space resembles the structure of foveated volume, with higher density around a fovea, our algorithm is a possible solution to maximize the efficiency of the system.

## 2.2 Proposed method

As mentioned in Chapter 1, we are interested in rendering the foveated volume directly from its received wavelet coefficients. We want to render the volume using the volume rendering E.q. 2.2.

Now our goal is to find an algorithm that can directly render the foveated volume from the relevant coefficients. We employ the notion of foveation to achieve different levels-of-resolution for volume rendering.

A foveated image can be viewed as a non-uniform sampled image, where the density of samples is the highest at the fovea, but falls off as the distance from the fovea increases. Figure 2.1 shows an example of a foveated image. Compared with Figure 2.1

(a) Uniform resolution image. (b) Foveated image.

Figure 2.1: Foveation.

(a) which is a full resolution image, the foveated image in Figure 2.1 (b) has a space variant resolution.

Similarly, a foveated volume can be viewed as a blending of multiple regions, each with a different level of resolution. By exploiting the relevant wavelet coefficients, a fast volume rendering can be achieved. The running time is $O(n^2 + m)$, where $n$ is the width of the rendered image, and $m$ is the number of wavelet coefficients retained for the foveated volume.

The proposed algorithm consists of two phases. The first phase is a fast reconstruction of the *super-voxels* from the wavelet coefficients, and the second phase renders the super-voxels by carefully tracking rays with different thickness in the super-voxels.

Previously known fast rendering algorithms do not fully exploit the information reduction in the sense that, voxels that appear before or after the ROI are either omitted or rendered in high resolution. Our algorithm achieves speedup by tracking the "thickness" of the rays during rendering. There is no expensive preprocessing on the wavelet coefficients. Hence, it is possible to interactively modify different viewing parameters such as the transfer functions.

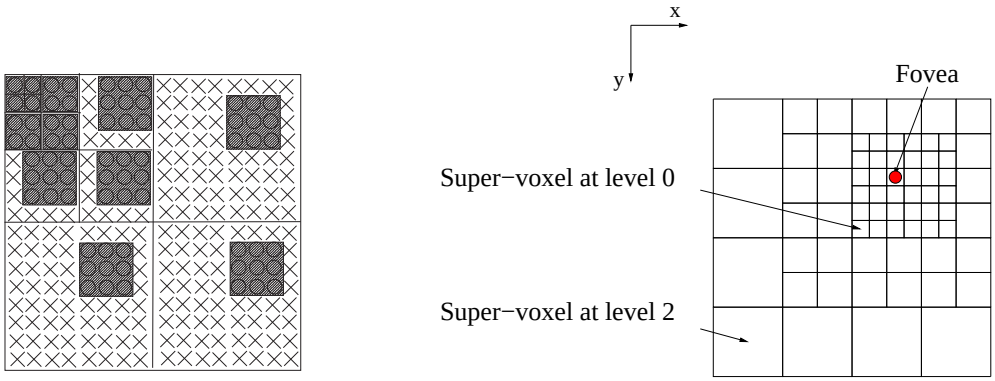



Figure 2.2: Wavelet foveation. (a) $\mathbf{x}_0 = (10, 4), r_0 = 3$ in wavelet domain. (b) $\mathbf{x}_0 = (10, 4), r_0 = 3$ in spatial domain.

### 2.2.1 Representation of foveated volume

We use a simplified approximation which is called *0-1 mask* [15] for the foveation process to get a foveated volume. This process depends on two parameters, the *fovea* $x_0$ which is a point in the 3D space, and *rate* $r_0$, a non-negative number. Given a volume $V$, foveation applies space-variant smoothing function on $V$. At locations nearer to the fovea, the width of the smoothing function is smaller. The rate $r_0$ determines how fast the width of the smoothing function grows.

Here describes the foveation process using a $16 \times 16$ pixels image. A similar idea can be applied on volume. Figure 2.2 (a) shows the retained coefficients for a foveated image. Coefficients in the shaded squares of Figure 2.2 (a) are retained. If the image is represented using Haar wavelet, then the foveated image is as shown in Figure 2.2 (b) where pixels in each box have the same value. Note that the widths of the shaded squares are the same except for those that touch the boundary. The location of each square with respect to the co-ordinate of the sub-band depends on the fovea location $\mathbf{x}_0$. The common width of the squares depends on the rate $r_0$. For convenience, we simply refer the width as rate. A better approximation can be achieved by using circles instead of squares, applying a weighting function on the coefficients, and having circles with slightly different size in different sub-bands [16].

**Co-ordinate system.** Our volume data-set $V$ is stored in a $n \times n \times n$ array. The indices of the array (starting from 0 to $n-1$) also serve as the locations of the voxels in the 3D space.

Same as in images, a wavelet coefficient of the three dimensional $V$ is labeled by its sub-band and location. Unlike images, in three dimensions, there are seven high frequency sub-bands at each level. We use the convention that sub-bands with the coarsest resolution are defined to be at the 0-th level. Thus, performing forward wavelet transformation on the $i$-th level sub-band $\mathtt{LLL}_i$ gives a $(i-1)$-th level sub-band $\mathtt{LLL}_{i-1}$, and seven other high frequency sub-bands. We also assume each sub-band is stored in a 3D array and use its index to serve as the location of the wavelet coefficient. Hence, the spatial location $(x, y, z)$ corresponds to $(x/4, y/4, z/4)$ in the sub-band $\mathtt{LLL}_{\log_2 n-2}$.

We call a coefficient in a low frequency sub-band $\mathtt{LLL}_\ell$ a *super-voxel* at level $\ell$. Each super-voxel can be viewed as a cube in the spatial domain. A $\ell$-th level coefficient at the location $(x, y, z)$ (with respect to the co-ordinate in the sub-band) corresponds to a cube of width $n/2^\ell$ at $(n/2^\ell x, n/2^\ell y, n/2^\ell z)$ in the spatial domain.

**Wavelet foveation and super-voxels.** For convenience, we simply call the approximation of the "ideal" foveated volume the foveated volume. Recall that the approximation is done by selectively retaining some coefficients, and is parameterized by the location of fovea $\mathbf{x}_0$, and the rate $r_0$. We denote the foveated volume data as $V_{\mathrm{f}}(\mathbf{x}_0, r_0)$.

Let $C(\ell, \mathbf{x}_0, r_0)$ be the set of wavelet coefficients in the $\ell$-th level high frequency sub-bands, and is contained in the cubes whose two opposite corners (with respect to the co-ordinate in the respective sub-band) are at

$$(n/2^\ell)\mathbf{x}_0 - \mathbf{r}', (n/2^\ell)\mathbf{x}_0 + \mathbf{r}' \tag{2.5}$$

where

$$\mathbf{r}' = \begin{pmatrix} \frac{r_0}{2} - 1 \\ \frac{r_0}{2} - 1 \\ \frac{r_0}{2} - 1 \end{pmatrix} and \ (r_0 >= 2).$$

The $C(\ell, \mathbf{x}_0, r_0)$ is in fact the $\ell$-th level of wavelet coefficients retained for the foveated volume $V_{\mathrm{f}}(\mathbf{x}_0, r_0)$. Let $\mathcal{C}(\mathbf{x}_0, r_0) = C(0, \mathbf{x}_0, r_0) \cup C(1, \mathbf{x}_0, r_0), \ldots, C(\log_2 n - 1, \mathbf{x}_0, r_0) \cup \{w_0\}$ where $w_0$ is the only coefficient in $\mathtt{LLL}_0$. Hence, from $\mathcal{C}(\mathbf{x}_0, r_0)$, we can obtain $V_{\mathrm{f}}(\mathbf{x}_0, r_0)$ using inverse wavelet transformation.

Consider the coefficients in the sub-band $\mathtt{LLL}_\ell$, and are within the cube with the two corners given by E.q. (2.5). Each coefficient is a super-voxel, and let us denote these coefficients as $R(\ell, \mathbf{x}_0, r_0)$. The foveated volume $V_{\mathrm{f}}(\mathbf{x}_0, r_0)$ can be obtained by merging the super-voxels in $R(0, \mathbf{x}_0, r_0)$, $R(1, \mathbf{x}_0, r_0), \ldots, R(\log_2 n - 1, \mathbf{x}_0, r_0)$. Note that the total number of super-voxels is same as the number of wavelet coefficients in $\mathcal{C}(\mathbf{x}_0, r_0)$.

### 2.2.2 Algorithm on rendering of foveated volume

Given the rate $r_0$, location of fovea $\mathbf{x}_0$, the wavelet coefficients $\mathcal{C}(\mathbf{x}_0, r_0)$ of the foveated volume, and the viewing parameters including the viewing direction $\theta$ and the transfer functions, we want to compute the rendered image of $V_{\mathrm{f}}(\mathbf{x}_0, r_0)$.

A straightforward algorithm solves the problem by first reconstructing the foveated volume $V_{\mathrm{f}}(\mathbf{x}_0, r_0)$ from $\mathcal{C}(\mathbf{x}_0, r_0)$ using inverse wavelet transformation, and next applying direct rendering on $V_{\mathrm{f}}(\mathbf{x}_0, r_0)$. This method is costly since representing $V_{\mathrm{f}}(\mathbf{x}_0, r_0)$ in the spatial domain already requires $\Omega(n^3)$ storage space. We give an algorithm that avoids reconstructing $V_{\mathrm{f}}(\mathbf{x}_0, r_0)$.

Our algorithm consists of two phases, reconstruction phase and rendering phase. In the first phase, given $m$ wavelet coefficients of the foveated volume, the super-voxels $R(\ell, \mathbf{x_0}, \mathbf{r_0})$ is reconstructed. The reconstruction can be done in $O(m)$ time. In the second phase, the displayed image is rendered from the super-voxels. The rendering time is $O(m + n^2)$. These two phases can be combined to further reduce memory usage.

**Rendering.** Let us first describe the second phase which is more interesting. We will explain the rendering using a 2D example. We want to trace rays in a foveated image along the x-axis as shown in Figure 2.2 (b), giving a 1D signal as output. In Figure 2.2 (b), a lower resolution sample is depicted as a bigger square, which we call it a *super-pixel* (the analogous of super-voxel). Consider a set of rays tracing through a big super-pixel. If the intensities of the rays are the same before hitting the square, then they are also the same upon leaving the square. Thus, from computational aspect, all these rays can be emulated altogether in one step. Since they are the same, we group these rays into a *thick* ray, where the thickness is the width of the region it covers.

A key observation is that we can always *split* a thick ray, but not mix two rays. Consider the situation where a thick ray leaves a square and enters into two smaller squares. In this situation, the ray has to be split into two thinner rays. On the other hand, consider the situation where two adjacent thin rays, leave their respective squares and enter into a common bigger square. In this situation, the two rays may be different in intensity, when entering into the bigger square. Hence, no computation can be shared.

The darker arrows in Figure 2.3 (a) show how the rays trace half-way through a foveated image. Due to the structure of foveation, we only need to split the rays. Problem arises in the second half of the foveated image if the rays continue to trace toward the right. Since rays can not be mixed, in the second half, they have to remain thin. This is not optimal since, intuitively, some computation could be shared in the second half. To overcome that, we trace the rays along x-axis in two directions, forward and backward as shown in Figure 2.3 (b). The final rendered 1D signal is the composition of these two sets of rays. We do not set the line where the two sets of rays meet as a straight line, otherwise it may cut across a whole square.

For arbitrary viewing directions, we first apply shear-warp [49] on the super-voxels, and perform geometric correction on the rendered image. To illustrate this process clearly, we give the detailed explanation with an example on rotation of a foveated image in Chapter 3. Similar idea can be easily extended on foveated volume.

**Reconstructing super-voxels.** Given the wavelet coefficients $\mathcal{C}(\mathbf{x}_0, r_0)$ of the foveated volume (the shaded squares in Figure 2.2 (a)), we want to reconstruct the super-voxels. A full inverse wavelet transform will be costly. Fortunately, due to the special arrangement of those coefficients, the reconstruction can be restricted within a cube of width $(r_0 + s)$, where $s$ is the wavelet support size. Thus the running time is in the same order as the number of selected coefficients. To further speedup, we can restrict reconstruction in the width of $r_0$, however, there will be a minor lost in accuracy.

There are two methods to reconstruct the super-voxels.

1. Before rendering, reconstruct the super-voxels from the wavelet coefficients. This can be done efficiently, with running time in the same order of the number of retained coefficients, and an additional memory space of the same order is required. A main advantage is that the reconstruction process and the rendering process are separated. Hence, a different reconstruction algorithm can be employed without changing the rendering algorithm. For example, we could represent the volume in any wavelet. As long as the super-voxels can be reconstructed efficiently, the rendering can proceed.

2. The super-voxel is reconstructed as required during rendering. In this method, less additional storage is required. However, since the reconstruction and rendering are to be performed together, the algorithm is more complicated. Hence, it is difficult to incorporated changes in the reconstruction.

In our implementation, we use the second method. However, for simplicity in explaining the rendering algorithm, we assume that the super-voxels have already being reconstructed.

**Total running time.** The reconstruction phase takes $O(m)$ time where $m$ is the number of wavelet coefficients. Also recall that the number of super-voxels is also in $O(m)$. For rendering, observe that the computation required is directly proportional to

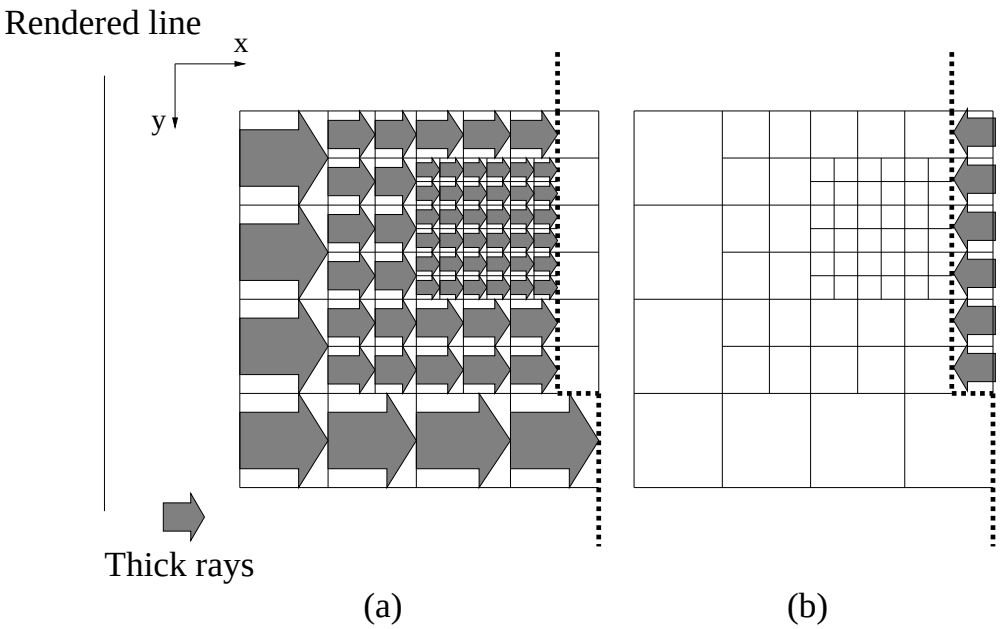



Figure 2.3: Thick rays rendering.

the number of rays, which is the number of super-voxels. Hence, the running time will be $O(m+n^2)$ where $m$ is the total number of wavelet coefficients required, and $n$ is the width of the rendered image.

**Combining reconstruction and rendering.** If the volume is represented using Haar wavelet, it is possible to combine the reconstruction phase and rendering phase, so that the super-voxels are computed as and when required and not explicitly stored. In this way, additional memory space required can be reduced. In our implementation, we combine these two phases.

**Using wavelets with a larger support.** As mentioned in Section 2.2.2, the rendering algorithm essentially works on the reconstructed super-voxels $R(\ell, \mathbf{x}_0, \mathbf{r}_0)$ for $0 \leq \ell < \log_2 n$, and the reconstruction process can be separated from the rendering. Hence, it is possible to represent the volume using wavelets with larger support, for example Daubechies 7/9 biorthogonal wavelets. What is required is an efficient reconstruction algorithm that obtains the super-voxels from the wavelet coefficients. Due to

the special structure in foveated volumes, we do not need to perform the full inverse wavelet transformation to obtain the super-voxels. The reconstruction can be restricted within a cube of width $r_0$, and thus achieving running time in the same order as the number of selected coefficients. In contrast to the full inverse wavelet transformation, this fast reconstruction gives an approximation of the super-voxels. Nevertheless, the approximation is accurate.

**Rotation approximation in the wavelet domain.** For arbitrary viewing directions, we first apply shear-warp factorization [49] on the super-voxels, followed by geometric correction on the rendered image. The detailed explanation on this process is given in Chapter 3. For easy illustration, an example on rotating a foveated image is discussed. Similar idea can be applied on foveated volumes.

### 2.2.3 Visualizing foveated volume

A foveated volume implicitly indicates that the interesting features are near the fovea. Hence, for effective visualization, it is desirable to give priority to the fovea, and to have a mean to direct the viewer's attention to the fovea. This can be achieved by multiplying the original opacity with a space-variant *weighting function*. Specifically, the opacity at location $(x, y, z)$ is $T_\alpha(V(x, y, z))D_{\mathbf{x}_0}(x, y, z)$ where $V(x, y, z)$ is the voxel intensity, $T_\alpha(\cdot)$ is the opacity transfer function, $\mathbf{x}_0 = (x_0, y_0, z_0)$ is the fovea, and $D_{x_0}(\cdot)$ is the weighting function. Hence, the opacity of a voxel depends on both its location, and its intensity. We experiment with two weighting functions.

- The weighting function chops off all the voxels before the fovea along the viewing direction. If the viewing direction is along the x-axis, the function is:

$$D_{\mathbf{x}_0}(x, y, z) = \begin{cases} 1 & \text{if } x > x_0, \\ 0 & \text{otherwise} \end{cases}$$

- The weighting function varies across the 3D space. It is higher near the fovea, and

its reciprocal increases linearly as the distance from the fovea increases. Specifically,

$$D_{\mathbf{x}_0}(x, y, z) = (1 + a\|\mathbf{x}_0 - (x, y, z)\|_2)^{-1}$$

where $\|\cdot\|_2$ is the usual 2-norm and $a$ is a constant that can be interactively adjusted by the viewer. When $a$ is small, the variation across the space is lesser.

### 2.2.4 Post-processing by low pass filtering

The staircase artifacts (that is, the "blockish" effect) in the rendered image are due to the notion of thick ray in sharing computation. A way to reduce the artifacts is by post-processing. We can view the output of the rendering as a collection of non-uniformly spaced samples of thick rays, and the rendered image is the interpolation of these samples. The staircase artifacts appear when the sampling function is a step function. Alternatively we can use a smoother sampling function to reduce the artifacts. This can be done by performing a space-variant smoothing process on the original rendered image, where the width of the smoothing function is larger for thicker rays.

Figure 2.4 depicts the smoothing process on the rendered image. The 3D blocks represent the super-voxels in the foveated volume. The smoothing is performed by applying low pass filters on the pixels of the rendered image. As the rendered image has a variable-resolution nature, the size of low pass filters may not be the same on different level of resolutions. For example, the low pass filter $F_1$ is larger than $F_2$ when they are employed for the pixels represented by cross signs in Figure 2.4 (i). Considerations must be taken when the frontals of super-voxels with different size do not meet on the same plane, for example, the upper graph in Figure 2.4 (ii). In this case, the low pass filters should be revised as shown in Figure 2.4 (ii), i.e., filter $F_1$ will not take samples from super-pixel $B$ and vice versa for filter $F_2$.

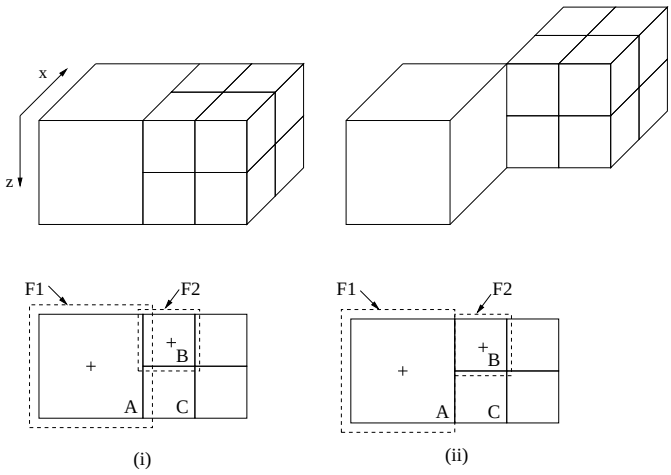



Figure 2.4: The post-processing by space-variant smoothing.

## 2.3 Implementation and experiments

In this Section, we give experimental analysis on our proposed algorithm. We inspect the performance of the algorithm by measuring the amount of wavelet coefficients required during foveated volume rendering and presenting visual effect with different foveation parameters. We also compare the algorithm with some existing volume rendering library. Finally, we describe some potential applications by applying our algorithm.

### 2.3.1 Experimental data-sets

To evaluate the visual effect of our proposed algorithm, we applied our method on the CT scan of the visible man's torso with 512x512x426 voxels. To evaluate the performance of our algorithm, we tested our method on three data-sets: a MRI scan of a head with 128x128x84 voxels, a CT scan of an engine with 256x256x110 voxels, a CT scan of a human head with 256x256x225 voxels. These are described as "brain", "engine" and "head" in Table 2.1.

### 2.3.2 Experimental results

All experiments were conducted on a 3GHz Pentium IV PC with 1GB DDR RAM.

Figure 2.5: Rendering of a full resolution volume having 512x512x426 voxels with viewing angle $\theta = 30$ degree.

**Rendering results.** Figure 2.5 shows the rendering on the full resolution volume as the ground truth. The rendering results of foveated volumes are given in Figure 2.6 and Figure 2.7, all at a viewing angle of 30 degree.

Figure 2.6 (a), Figure 2.6 (c) and Figure 2.6 (e) show the rendering with fovea parameters $\mathbf{x}_0 = (155, 353, 300)$ and rate $r_0$=100, 50 and 25. The fovea is marked as a red dot. The number of coefficients retained for the foveated volume with rate 100, 50 and 25 is approximately 23.7x$10^6$, 5.6x$10^6$ and 0.9x$10^6$ respectively. This amounts to a reduction to 21.3%, 5% and 1% of the original volume. Note there are staircase artifacts around the peripheral. The artifacts are reduced after a space-variant smoothing is applied as shown in Figure 2.6 (b), Figure 2.6 (d) and Figure 2.6 (f).

Figure 2.7 (a), Figure 2.7 (c) and Figure 2.7 (e) show rendering results on the same data-set as Figure 2.6 (a), Figure 2.6 (c) and Figure 2.6 (e), except that the fovea is moved to another location.

Comparing Figure 2.5 and Figure 2.6, it is noted that no information is lost at the fovea while a large amount of coefficients are omitted for rendering foveated volumes.

Figure 2.8 shows the rendering results of applying weighting function mentioned in

Section 2.2.3. After applying the second weighting function on Figure 2.6 (c), we have Figure 2.8 (a). Compared with the one without weighting function, the peripheral region appears darker. This is because the weighting function further suppresses information far from the fovea. Figure 2.8 (c) employs a larger weighting parameter $a$ than Figure 2.8 (a). By this adjustment, the information about peripheral region is much more suppressed. Figure 2.8 (e) gives result when the rate $r_0 = 25$. To reduce staircase artifacts, Figure 2.8 (b), Figure 2.8 (d) and Figure 2.8 (f) apply smoothing on Figure 2.8 (a), Figure 2.8 (c) and Figure 2.8 (e).

Figure 2.9 (a) and Figure 2.9 (b) show the chopping off effect- the first weighting function mentioned in Section 2.2.3. The full resolution volume for Figure 2.9 (a) and Figure 2.9 (b) is 256x256x225 voxels and the rate $r_0$ is 40. Note that each of these images is not simply an image of a plane slicing through the volume. This can be observed in Figure 2.9 (b), where the surface of the ear is vaguely visible.

**Computational Performance.** Figure 2.10 gives rendering time for different viewing parameters and data width.

Figure 2.10 (a) shows that the rendering time increases as the fovea rate $r_0$ increases. It is because as $r_0$ increases, more wavelet coefficients are selected for rendering. For the same $r_0$, the time for rendering with viewing angle at 45 degree is larger than that at 0 degree since there are more data to be processed in the shear-warp operation. The reason is also true for Figure 2.10 (b) which shows that more computational time is required for larger angles. The worst case performance occurred at 45 degree. Here the rate is 40. The original volume for Figure 2.10 (a) and Figure 2.10 (b) has 512x512x426 voxels.

Figure 2.10 (c) shows that the rendering time increases as the data width increases. When $n$ is large, the time is proportional to $n^2$ as $m$ is small. When the width increases from $n = 1024$ to 4096, although the data size increases by a factor of $4^3 = 64$, the rendering time only increases by approximately a factor of 6.86 when the viewing angle $\theta$ is 45 degree.

Table 2.1: Comparison of frame rates on different data-sets. The viewing direction is along x-axis for direct volume rendering. Note that VolPack requires large preprocessing time. Due to the memory limit of our machine, we only compare these three methods on these small size data-sets. In Figure 2.10 (c), we give the performance analysis of our algorithm on larger data-sets.

| Data-set | Direct volume rendering (Frame rate) | VolPack (Frame rate/MV/MO/CV) | Our alg. (Frame rate) |
|---|---|---|---|
| brain | 32.3 | 106.4/0.91 0.03/0.16 | 43.5 |
| engine | 6.4 | 45.9/4.66 0.19/0.69 | 42.6 |
| head | 3.2 | 16.0/9.09 0.33/1.67 | 25.6 |

### 2.3.3 Comparison with other methods

We compared our proposed algorithm with the VolPack volume rendering library [1], and a straightforward direct volume rendering. Table 2.1 gives the frame rate (in Hz) and rendering time (in seconds) on the test data-sets by different rendering methods. Note that the large 512x512x426 voxels volume is not tested on VolPack since VolPack is unable to process the large volume under our machine configuration.

In direct volume rendering, the rendering equation 2.2 is applied to the full resolution volume using the straightforward for-loops, with the viewing direction along the x-axis.

There are three rendering algorithms provided by VolPack. The fastest algorithm relies on a special data structure containing run-length encoded, classified volume data. Preprocessing is required to obtain this data structure. Hence, it is suitable for rendering the same volume without changing classification. MV, MO and CV represent the three preprocessing steps, which are:

- Make volume (MV): Create an unclassified volume from the raw volume data. This unclassified volume includes precomputed information for shading and classification;
- Make octree (MO): Create a min-max octree from the unclassified volume;

- Classify volume (CV): Create a classified volume including an opacity with each voxel along with shading information.

VolPack provides accurate classification. Even if we just consider MO which deals with the octree and the structure of resolution, the preprocessing time is still non-negligible.

If the volume is already represented by its wavelet coefficients, no preprocessing is required for our algorithm. Hence, if the viewer wishes to interactively change the transfer functions for the intensity and opacity, our algorithm is still able to give real time feedback for large data-set. The foveation parameters we use in Table 2.1 are $r_0$=25, $\theta = 45$ degree.

## 2.4 Remarks

In this work, we presented an algorithm that renders a foveated volume efficiently in the wavelet domain. The required running time for rendering the foveated volume is $O(n^2 + m)$ where $n$ is the width of the rendered image, and $m$ is the number of retained wavelet coefficients. We implemented the algorithm and analyzed its performance. The experimental study also confirmed the efficiency of the algorithm, even for very large $n$. Excluding the forward wavelet transformation, no expensive preprocessing is required on the original volume. Compared to the rendering of the full resolution volume, our method produces the image with the same quality at the fovea but lower resolution further way. The method provides a good tradeoff between rendering resolution and frame rate. It is suitable to be applied in scenarios where the rendering platform has low computing resources and/or real time feedback is required.

### 2.4.1 Combining reconstruction and rendering

Note that in our implementation, we combined the two phases of reconstruction and rendering. An advantage is that it achieves further speed-up. However, there are some disadvantages. The implementation would not be easy, especially when shear-warping

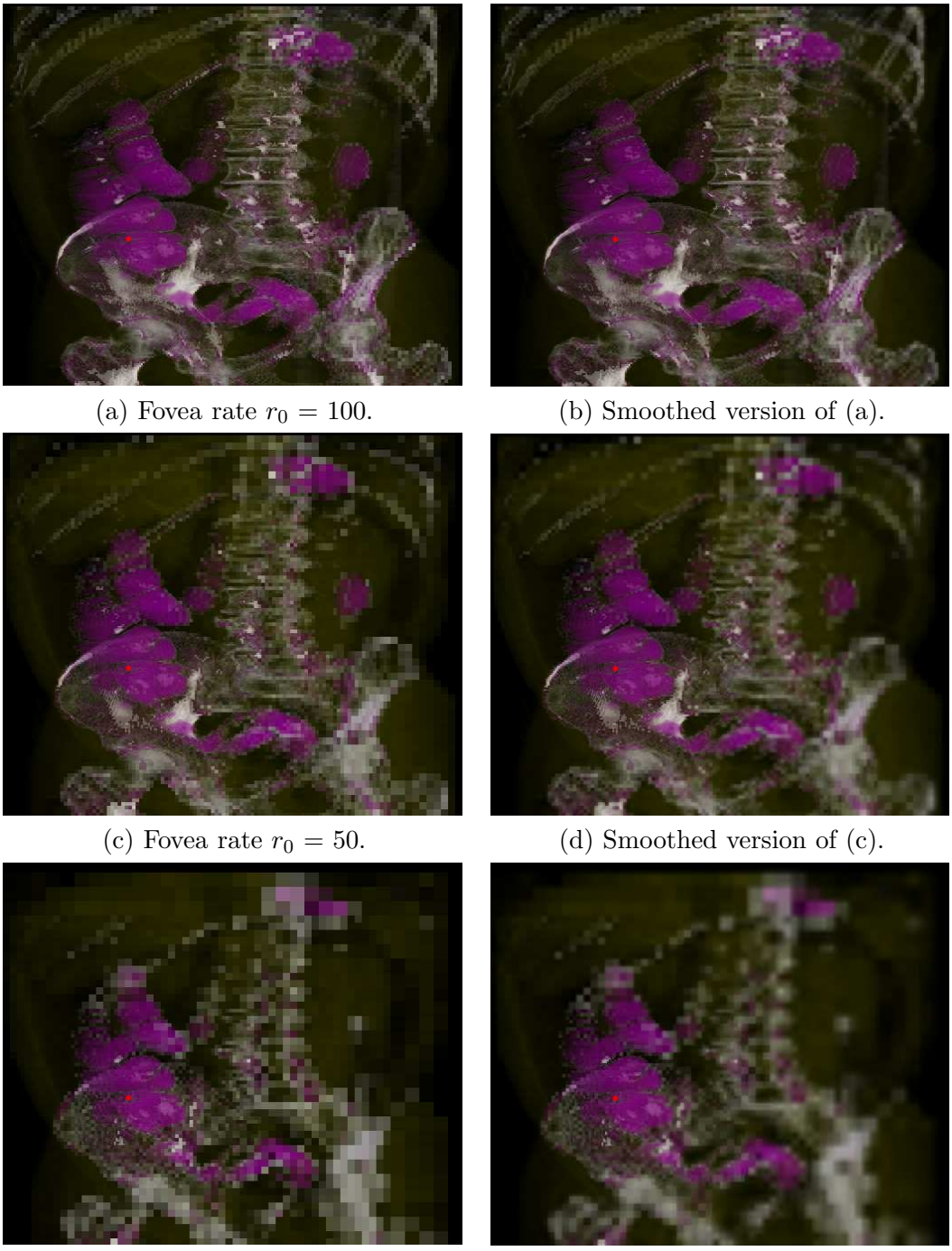
is involved. Furthermore, it also removes some flexibilities. For instance, it is not clear how to extend it to wavelet with larger support when both phases are combined. In addition, in applications where the volume has to be rendered many times with the same coefficients but different viewing parameters, the reconstruction will be unnecessary repeatedly performed if the reconstruction and rendering phases are closely coupled.

### 2.4.2 Future work

The current research work can be extended in the following two directions:

1. **Time varying foveated volume visualization.** The efficiency of our proposed method lies in the fact that the computational overhead is proportional to the number of relevant wavelet coefficients. However the problem becomes complex if the data are time dependent as the wavelet coefficients will change correspondingly. It is prohibitive to apply wavelet transform on volume data in the time domain. Further work will focus on how to correlate the wavelet coefficients of time series data thus reducing the rendering time.

2. **Foveated volume visualization with multiple foveas.** The extension to multiple foveas is not trivial. The challenging issue is that the special structure for thick rays rendering in Figure 2.3 will be complicated. When the multiple foveas exist, consideration should be given on how to carefully track the super-voxels partitioned by them.



(a) Fovea rate $r_0 = 100$.

(b) Smoothed version of (a).

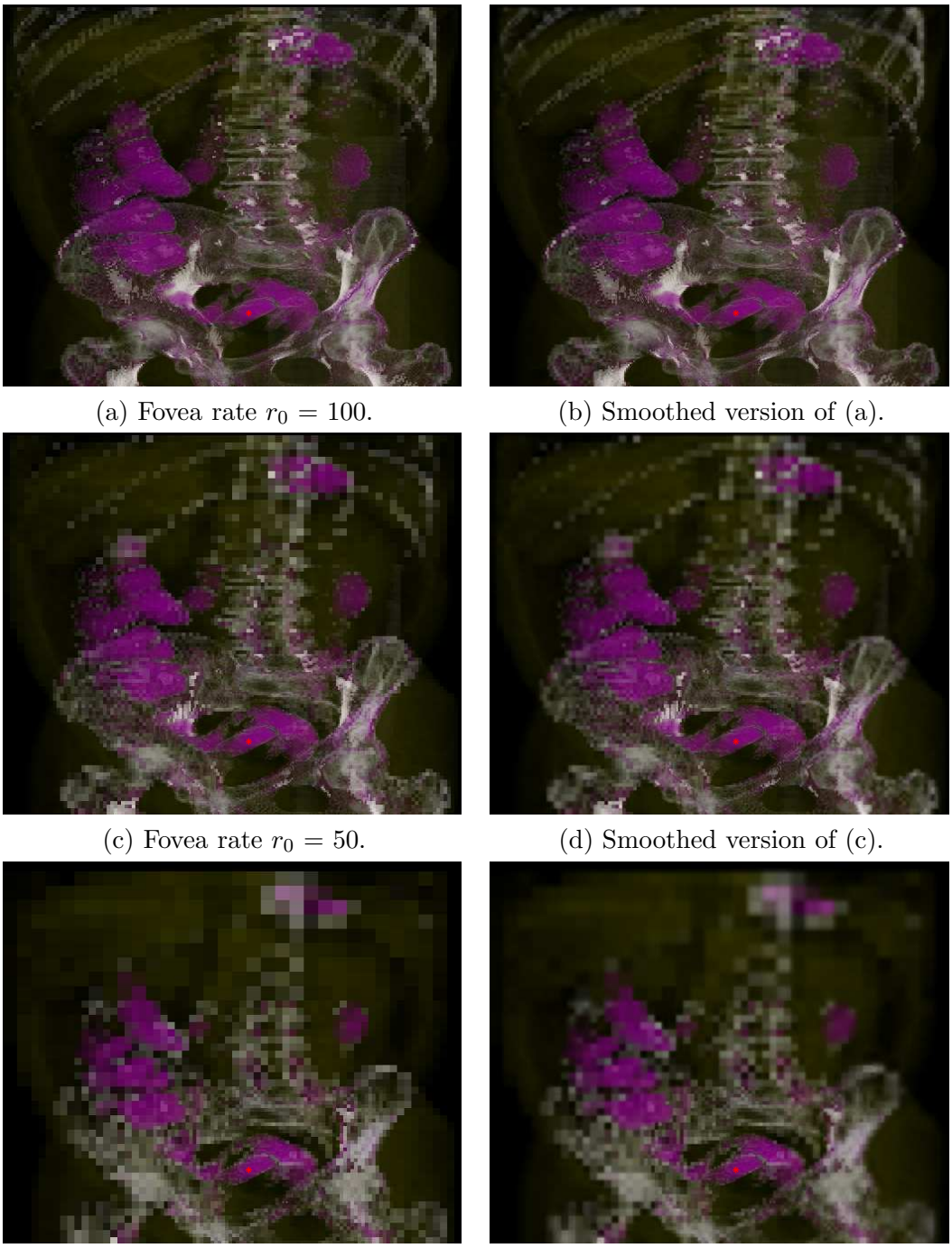
(c) Fovea rate $r_0 = 50$.

(d) Smoothed version of (c).

(e) Fovea rate $r_0 = 25$.

(f) Smoothed version of (e).

Figure 2.6: This set of images demonstrates the effect of fovea rate and location on the foveated volume. Each image in the right column is the smoothed version of the image at its left. The fovea is marked as a red dot in each image.



(a) Fovea rate $r_0 = 100$.

(b) Smoothed version of (a).

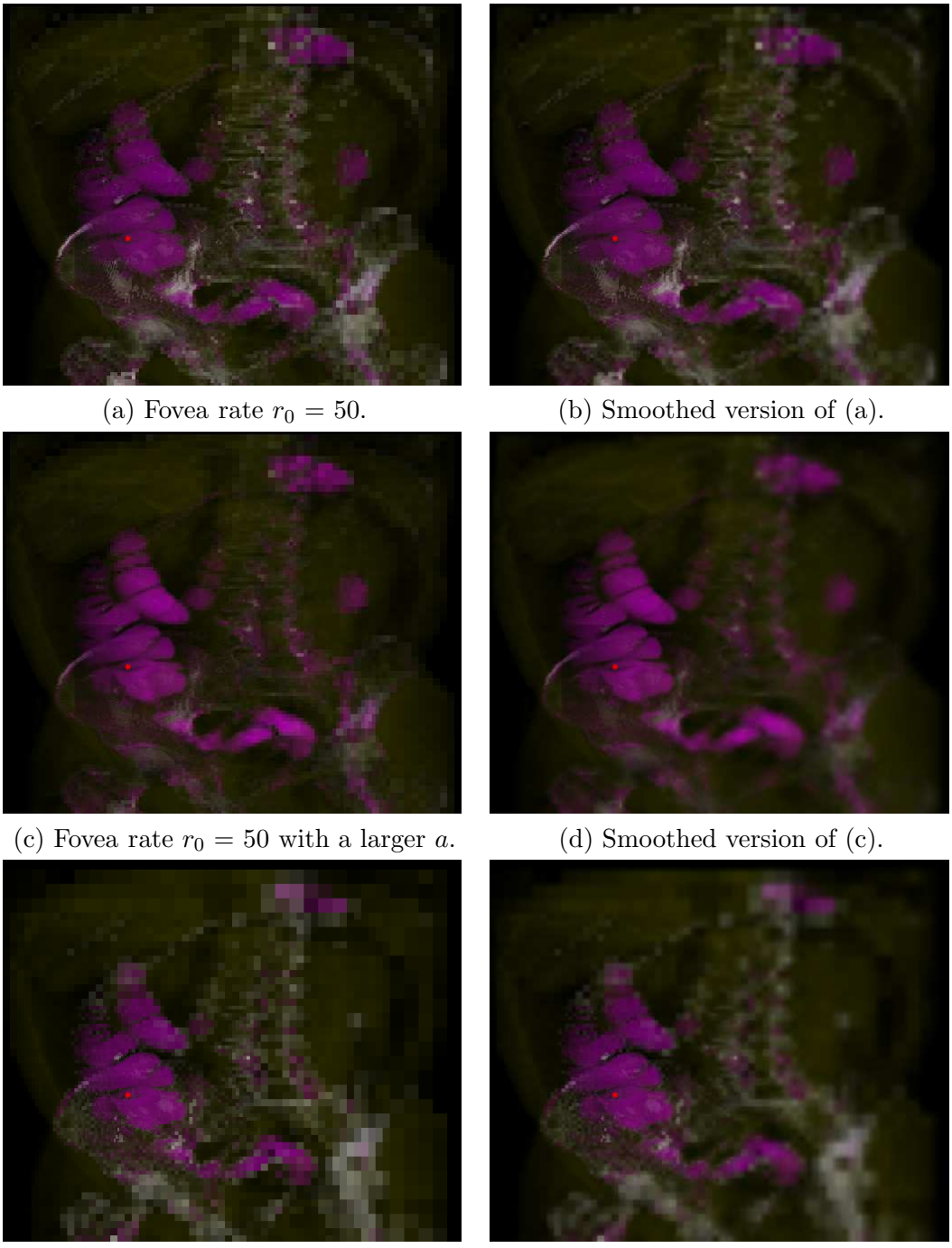
(c) Fovea rate $r_0 = 50$.

(d) Smoothed version of (c).

(e) Fovea rate $r_0 = 25$.

(f) Smoothed version of (e).

Figure 2.7: This set of images demonstrates the same effect as Figure 2.6 except a different fovea location. The fovea is marked as a red dot in each image.



(a) Fovea rate $r_0 = 50$.

(b) Smoothed version of (a).

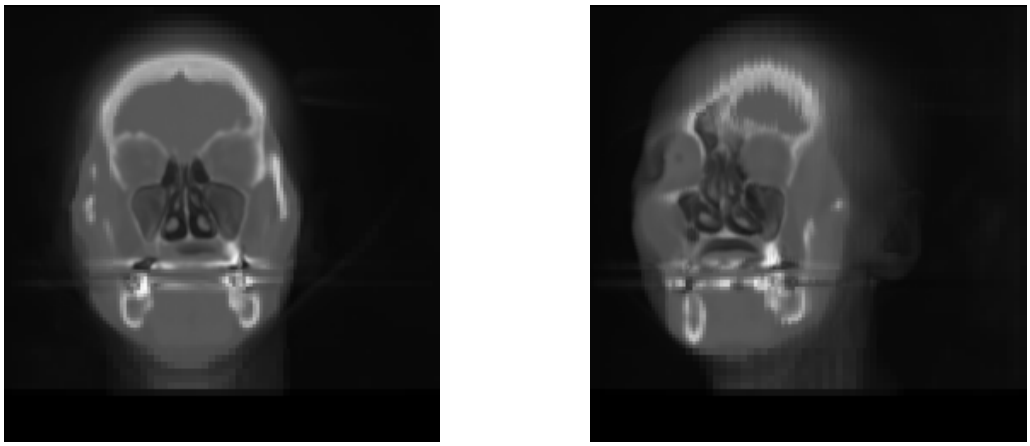
(c) Fovea rate $r_0 = 50$ with a larger $a$.

(d) Smoothed version of (c).

(e) Fovea rate $r_0 = 25$.

(f) Smoothed version of (e).

Figure 2.8: This set of images illustrates the effect of the second weighting function in visualizing foveated volume.



(a) (b)

Figure 2.9: This set of images illustrates the effect of the first weighting function in visualizing foveated volume. (a) The effect by chopping off the region before the fovea with viewing angle at 0 degree. (b) Same effect as (a) with viewing angle at 30 degree.

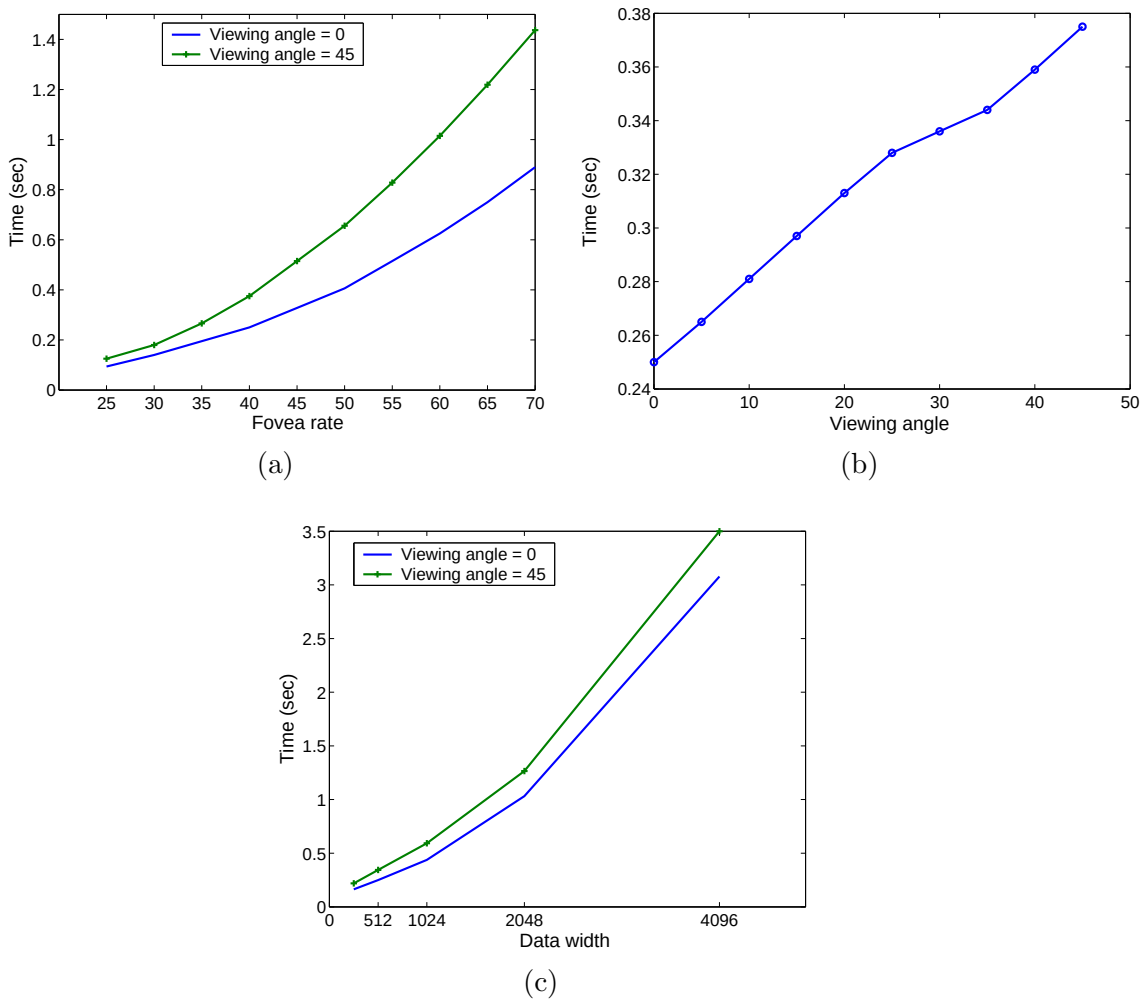



Figure 2.10: Rendering time. (a) Rendering time versus the rate $r_0$. (b) Rendering time versus viewing angle. (c) Rendering time versus data width. More time is required for a viewing angle at 45 degree.

# Chapter 3

# Rotation of foveated image/volume in the wavelet domain

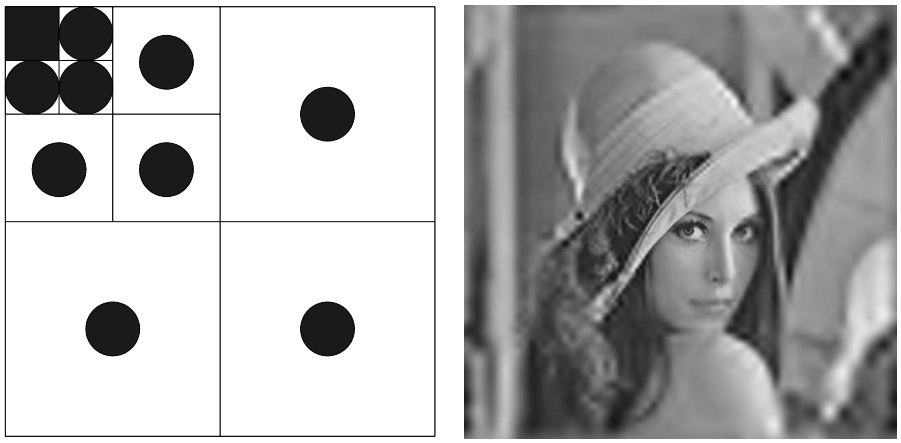
## 3.1 Introduction

This Chapter gives a side-result of the foveated volume rendering. The method to handle rendering in non-orthogonal direction could be extended to a more general result: How to rotate a foveated image/volume directly in the wavelet domain?

## 3.2 Proposed method

We give an approximation to obtain the wavelet coefficients of rotated foveated image/volume. The running time is proportional to the number of retained coefficients of the foveated image/volume. We propose two algorithms. The first algorithm is faster by a constant factor but the second algorithm is more accurate.

Figure 3.1 gives a foveated image represented in wavelet and spatial domain respectively. The foveated image is obtained by the *0-1 mask* operation mentioned in Section



(a) (b)

Figure 3.1: (a) the mask when the fovea is at the center; (b) foveated image whose mask has radius of 30 pixels and locates at center.

2.2.1. Figure 3.1 (a) shows the locations of the retained coefficients when the fovea is in the center and Figure 3.1 (b) shows the foveated image.

**The problem.** Consider a foveated image $I_1$ and its rotated version $I_2$. The input of our problem is $W_1$, the wavelet coefficients of $I_1$, and the output is $W_2$, an approximation of the wavelet coefficients of $I_2$.

A direct method requires three steps: (i) reconstructing the foveated image $I_1$ from $W_1$, i.e. applying IDWT on $W_1$, (ii) rotating the image $I_1$ to obtain $I_2$ and (iii) applying DWT on $I_2$ to get $W_2$.

Assuming that the image size is $n \times n$ pixels, each of the above steps requires a running time in the order of $\Theta(n^2)$, which is considerable when the image is large. This is especially so when higher order of interpolation is used during rotation. Recall that $W_1$ can be represented by small number of coefficients. Let $m$ be the number of coefficients retained during the foveation. In Figure 3.1, $m$ is the number of 1's in the mask. We want to design an algorithm that depends only on $m$.

Our algorithms are based on the observation that for a foveated image, the radius of the circles in the mask are the same across every level. Furthermore, the centers of the

circles all correspond to the same location, which is the fovea, in the spatial domain. Consequently, we can focus the operation on the area around the fovea and significantly reduce the running time.

The two algorithms are illustrated in Figure 3.3. Below are the detailed descriptions.

**Algorithm 1.** Consider the sub-bands in the wavelet transform. We call the $i$-th level LH, HL and HH high frequency sub-band $h_i$ (see Figure 3.2 (a)), and $l_i$ the LL sub-band of the $i$-th level. Thus $l_3$ can be reconstructed from $l_1$, $h_1$ and $h_2$. For a foveated image, its non-zero coefficients are concentrated in the arrangement of circles of radius $r$ shown in Figure 3.1 (a). We write $h_{i,r}$ as the coefficients retained in the sub-bands $h_i$. Thus, $h_{i,r}$ contains coefficients in 3 circles of radius $r$, where each circle corresponds to the LH, HL and HH sub-band.

Figure 3.3 shows the steps of Algorithm 1 when the number of levels is 3. It is easy to generalize to any number of levels. In this figure, $mask(r_0)$ is an operation that applies a mask of radius $r_0$ in the sub-band. For example, after $l_3$ is applied a $mask(2r)$, coefficients at a distance greater than $2r$ from the fovea will be removed (or equivalently, set to zeros). The operation "rotate" is the usual image rotation. We are not concerned with the interpolation method used during rotation. In our experiment, we employ bicubic interpolation.

At level 1, the inputs of Algorithm 1 are $l_1$ and $h_1$ of the foveated image. We apply IDWT on $l_1$ and $h_1$ to get $l_2$. Since the energy of $l_2$ is concentrated in the circle of radius $2r$, to achieve speed-up, a circular mask of radius $2r$ is applied to obtain $l_{2,2r}$. Next, $l_{2,2r}$ is rotated to produce a rotated $K_2$, followed by DWT to give the sub-bands $L_1$ and $H_1$. Similarly, the energy of each of these sub-bands is concentrated in a circle of radius $r$. Hence, we apply a mask of radius $r$ on $H_1$ to produce $H_{1,r}$. At level 2, similar process is repeated, except that the input is $l_{2,r}$. Note that masking is performed a few times in the above steps. Although masking reduces accuracy, it is necessary to make sure that the data size is small. After all, the values of the discarded coefficients are small. This recursive process is carried out until reaching the final level.

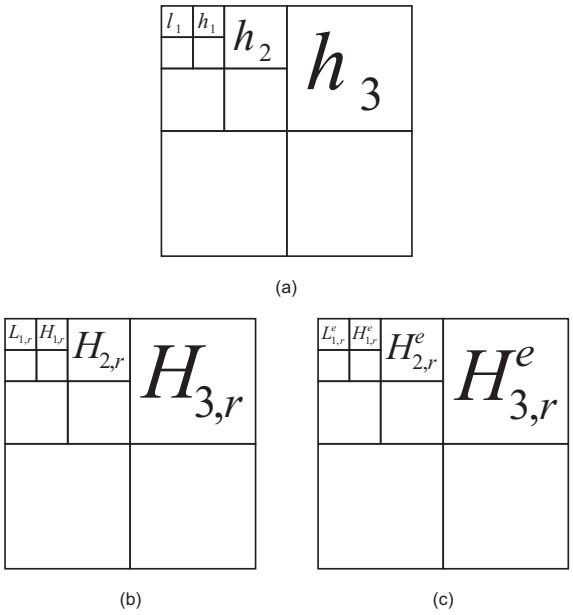



Figure 3.2: Wavelet transforms of: (a) original foveated image; (b) foveated image rotated by Algorithm 1; (c) foveated image rotated by Algorithm 2.

Note that each of DWT, mask(), and rotations can be done in $\Theta(r^2)$ time and there are only $N$ levels. Thus the total time required is $\Theta(Nr^2)$, which is $\Theta(m)$.

**Algorithm 2.** Algorithm 2 is a modified version of Algorithm 1. The main observation is the following. Let $K_{i,r}$ be the coefficients obtained by masking $K_i$ with the circle of radius $r$. Ideally, $K_{i,r}$ should be the same as $L_i$. However, this is not the case due to the combined effects of rotation and wavelet transformation. $L_i$ is more accurate because it is computed in the higher level. Therefore, if we retain $L_{i-1}$, $H_{i-1}$ which are calculated from $K_i$, in the final output, the LL sub-band at level $i$ will be the imprecise $K_i$.

In Algorithm 2, we introduce a step after $L_i$ is obtained. We first compute the error $E_i = L_i - K_{i,r}$. Next, DWT is applied on $E_i$. The wavelet transform of $E_i$ is then added to $L_1$, $H_1$, $H_2$, ..., $H_{i-1}$. To ensure $\Theta(m)$ computation, we approximate the DWT by restricting the coefficients in each sub-band to be in the circle of radius $r$.

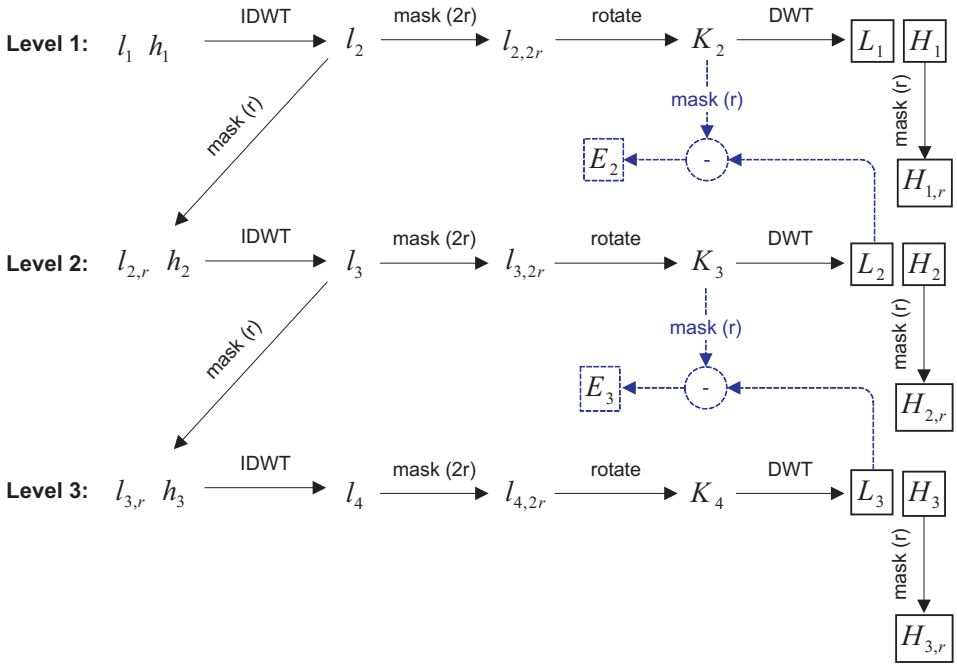
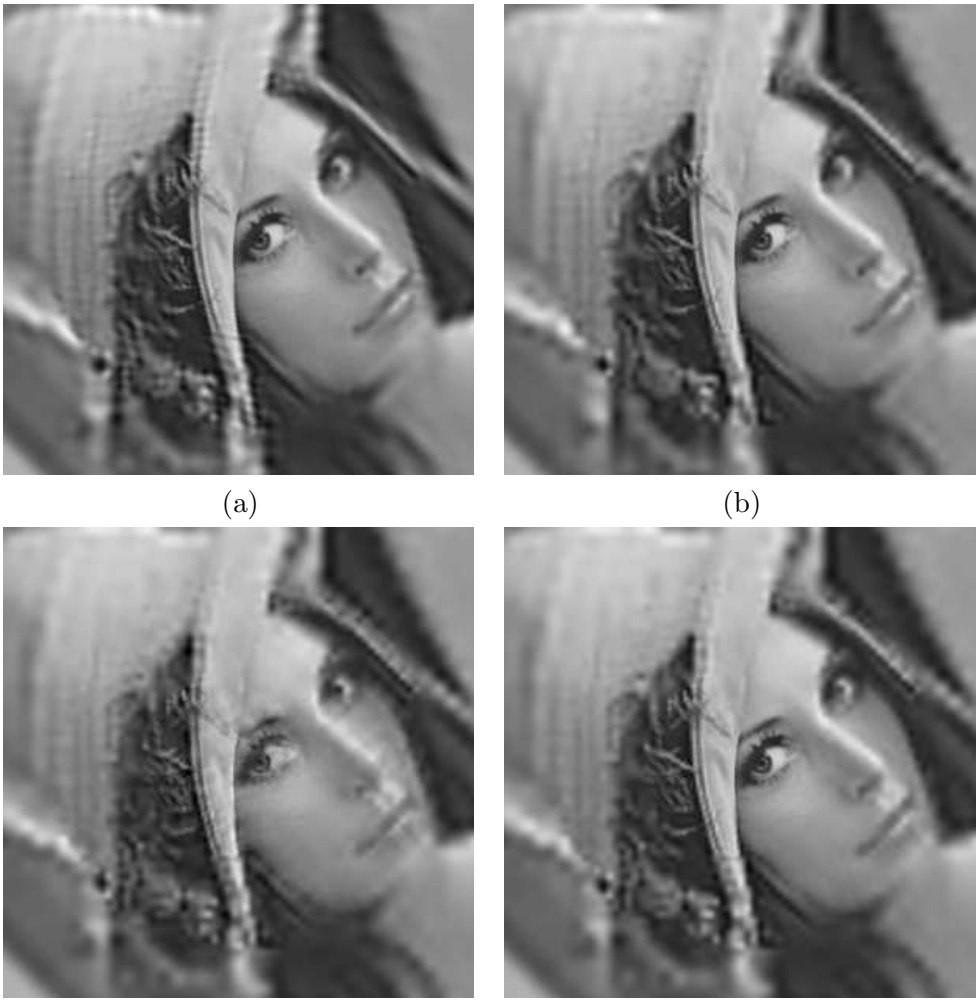



Figure 3.3: Algorithm 1 to rotate foveated image directly in wavelet domain (dotted lines are additional steps in Algorithm 2).

## 3.3 Experimental results

To measure the performance of the algorithm to approximate rotation in the wavelet domain, we implement our proposed algorithms to compare with traditional methods using several different images: "Lena" (512x512), "Mandrill" (512x512), "Cameraman" (256x256), "Peppers" (256x256) and "Barbara" (256x256). The performance of accuracy achieved is measured by Normalized Mean Square Error (NMSE). The wavelet filter used is the biorthogonal 7/11.

Figure 3.4 shows the rotated image by different methods when the degree of rotation is 45, and the radius of the mask is 15. To clearly show the differences between images, we present their zoom-in versions whose focuses are around the fovea. We denote $J_1$ the image obtained by directly rotating the foveated image $I_1$ in the spatial domain, which is shown in Figure 3.4 (a). We denote $J_2$ the image obtained by a straightforward algorithm: applying DWT on $J_1$, *mask* on the wavelet coefficients and IDWT on the wavelet coefficients in the mask. The zoom-in of $J_2$ is shown in Figure 3.4 (b). We

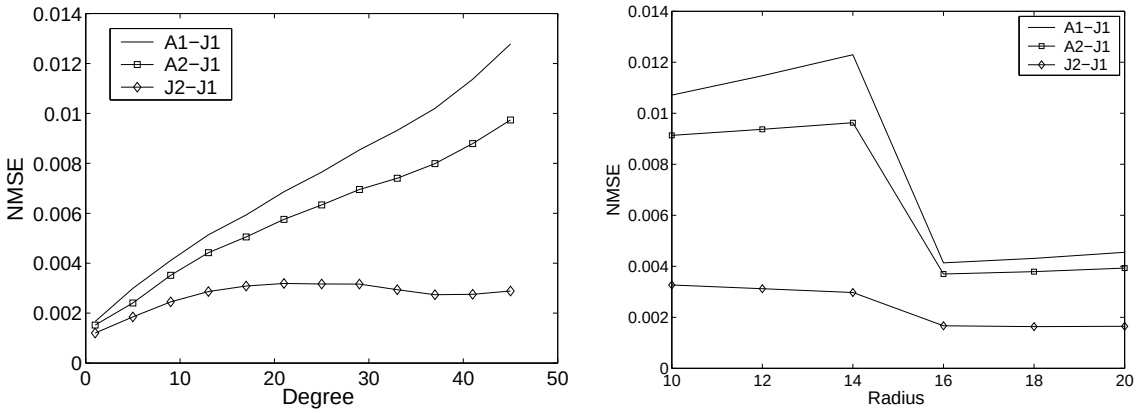


(a) (b)

(c) (d)

Figure 3.4: Rotated images: (a) $J_1$ obtained by rotating the foveated image Figure 3.1 (b); (b) $J_2$ obtained by applying DWT on $J_1$ and IDWT on the coefficients in the mask; (c) $A_1$ obtained by Algorithm 1; (d) $A_2$ obtained by Algorithm 2.

denote $A_1$ and $A_2$ the images obtained by our proposed Algorithm 1 and Algorithm 2, respectively. The zoom-in of $A_1$ and $A_2$ are shown in Figure 3.4 (c) and Figure 3.4 (d). It is obvious that the image $A_2$ is more accurate than the image $A_1$ comparable to $J_1$.




(a) (b)

Figure 3.5: Performance ratios: (a) NMSE as the rotating degree increases; (b) NMSE as the mask's radius increases.

To compare the differences among $A_1$, $A_2$, $J_1$ and $J_2$, we use Normalized Mean Square Error (NMSE), which is the mean square error normalized by the energy of $J_1$. Figure 3.5 (a) shows the NMSE as the rotation degree increases and Figure 3.5 (b) shows the NMSE as the radius of the mask increases. Because of the rotation and wavelet transform, there may exist spike values on the boundaries of rotated images, which lead to inaccurate NMSE values. Therefore, we ignore the image boundary by comparing only the region within the inscribed circle of the image boundaries. The displayed NMSE values are the average results of 5 different images which we mentioned above.

We can see that the NMSE increases as the rotation degree increases, and decreases as the radius of the mask increases. It is because the larger the rotating degree is, the larger the number of non-zero coefficients are created in wavelet domain. It means that while the radius is unchanged, the number of non-zero coefficients being discarded (by our algorithms) increases and consequently NMSE increases. Similarly, when the radius increases, more number of coefficients are retained and we obtain more accurate

results. The NMSE difference shown in Figure 3.5 also confirms what we have visually concluded from Figure 3.4 about the accuracy of the proposed algorithms. Note that the graph of $(J_2 - J_1)$ indicates the best approximation one can achieve.

## 3.4 Remarks

Although wavelet transforms have been studied for more than a decade, there are few researches on direct and efficient manipulation in the wavelet domain. In this chapter, we propose two algorithms that directly rotate foveated images in the wavelet-based compressed domain. The running time of our algorithms depends only on the number of retained coefficients, and this is considerably faster than the straightforward algorithm. While the proposed approximation algorithms are more efficient, experimental result shows that the approximation is also accurate. Here is a possible deployment of our algorithms: Suppose a method can efficiently extract features of foveated images directly from its wavelet coefficients, and we want to extend it to rotated images. This extension can be done by first applying our algorithm, followed by the original efficient extraction method. In the future, it would be interesting to explore similar techniques for other operations.

# Chapter 4

# Vector map visualization using region-of-interest

## 4.1 Introduction and related work

A Geographic Information Systems (GIS) is a collection of software tools that is responsible for gathering, organizing, analysing, manipulating and presenting geospatial data and related information. The role of GIS has greatly affected all aspects of human activities, such as agriculture, forestry, military, transportation and urban planning etc. The power of GIS lies in the fact that by the integration with database capabilities it creates a link between visual representation of geographic features and their existences in the database. This connection provides much convenient for answering complex queries issued by GIS users and helps GIS being a valuable tool in the process of decision-making operations.

In geographic visualization, there are two fundamental formats to represent spatial objects: vector- and raster- based. Vector-based format stores spatial objects by their coordinates and the connection among these coordinates. It is most appropriate for modeling linear features, such as roads, and rivers. The advantage of this data type is that it will not be distorted when it is viewed at different scales. Raster-based format

stores spatial objects by an image. It is suitable for some spatial operations, such as map overlays or area calculations. The data format in this thesis is vector-based as it can achieve better scalability than raster-based format. If the vector map is stored at the largest scale, i.e. the highest resolution, it can be used repeatedly by multiple users with different desired scales through map generalization.

With the development of mobile devices, there emerged some applications targeted on these devices, such as Location-Based Services and Global Positioning Services. Although the mobility of these applications provides much convenience in personal navigation, there are a few disadvantages in the practice. Generally, these devices have restricted computing resources and limited display windows thus making it difficult to process and provide visualization of large data-set.

In this Chapter, we consider visualization of vector-based map in a small display window. It is more effective to provide multiple scales of geographical information to the viewer, with finer scale at the point of interest. With a small display window, a natural solution is to merge the variable scaled geographical information into a single image. Hence, the viewer is able to focus on the interesting region, while having a good grasp of the surrounding context. This is essentially visualizing the map through a fisheye lens. However, the fisheye lens induces undesirable geometric distortion in the peripherial, which renders the information meaningless. Our solution is to apply map generalization that removes excessive information around the peripheral and a smoothing process to correct the distortion.

### 4.1.1 Variable-scale display techniques on vector map

Browsing map based on variable-scale display techniques can be classified into three categories [35]:

1. The map at different scales is obtained by zooming plus panning operation. This is the basic function for most GIS. Similar idea is applied to display large graphs by a single view [7]. When the zooming is performed, there are several different ways

to retrieve data: the data are displayed in a larger or smaller area; based on the former choice, the data are enriched with more details for zooming in. Although this method is simple to realize, the disadvantage is that the viewer may have difficulties to mentally connect information from maps at different scales.

2. The map is displayed in multiple windows by overview plus detail visualizations [39]. For example, an overview window shows the overall map while some detail windows give map view at different scales. Viewers can have a simultaneous view of these multiple windows and switch among them. However there may exist overlap among the windows thus some objects may be occluded.

3. The variable-scale representation of the GIS data is displayed in the same map. Usually the regions near the location of viewers are shown in a larger scale and those far away are shown in a decreased scale. It is also known as focus plus context techniques [45] where the focus is the point of interest of viewers and the context supplies the whole picture. By this means, the information from different scopes is fitted together as a single event for viewers. Besides presenting large amount of information, it also provides much convenience for viewers to grasp both local and global information.

The research in this thesis is in the direction of the third approach mentioned above.

### 4.1.2 Variable-scale display techniques on logical data

Variable-scale techniques are not restricted to display vector map. They are also applied in many applications on visualizing logical data. Usually these techniques are called focus plus context approaches in the literature. Here we give a review on 5 typical techniques.

#### Fisheye View

Fisheye view was first introduced by Furnas [31]. A view like the shooting from a fisheye camera lens is created based on degree-of-interest function that calculates the

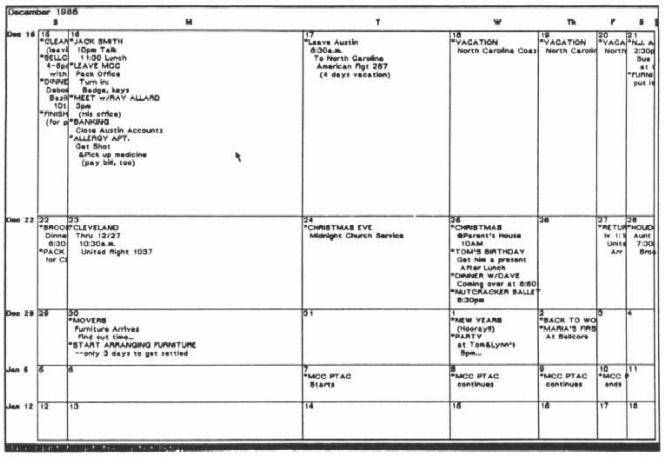


Figure 4.1: Fisheye view of a calendar in the work by Furnas [31] (Reproduced with permission of the author).

relevant scale of each object in the display space. The objects close to the focus are magnified while the rest are shrinked. Figure 4.1 gives an example of visualizing a calendar by fisheye view in Furnas's work.

According to the definition of the distance between an object and the focus, fisheye view can be divided into two classes: graphical fisheye view [87] and logical fisheye view [47]. On graphical fisheye view, the distance is measured by the Euclidean distance and for the latter one the distance is measured by some logical relationship among objects. Logical fisheye view is more rational to represent the structure of data however the layout may be changed drastically when the focus is adjusted.

### Perspective Wall

Perspective wall [61] was designed to visualize data with linearly structure such as chronological list of documents. This visualization technique maps 2D layout of data onto a 3D structure consisting of three walls. The focus view is presented in a frontal wall while the context view is displayed on the two side walls with decreasing magnification factor. Upon changing the focus, the walls are scrolled correspondingly. Figure 4.2 gives the perspective wall representation of a computer file that is listed by edit date and file type.

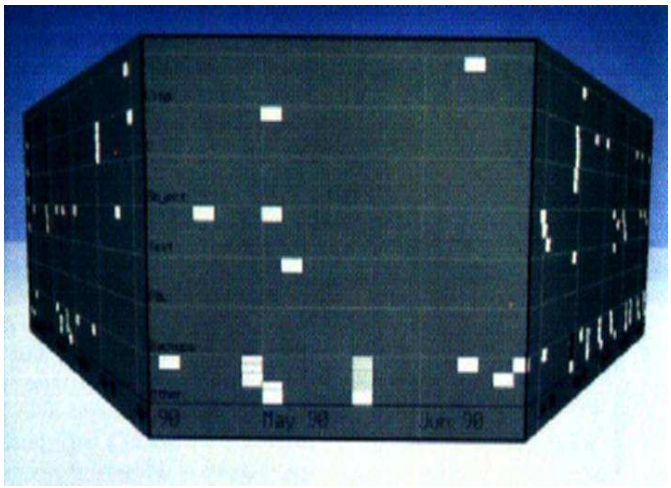


Figure 4.2: Perspective wall representation of a file in computer system in the work by Mackinlay et al. [61] (Reproduced with permission of the author).

### Bifocal Display

Bifocal display [94] has the similar idea as perspective wall that "folds" an information space to hold more data simultaneously. There are two steps to obtain bifocal display. The information space is wrapped around two uprights and projected on the screen space. The focus view between the two uprights is kept the same as the original while the other sides of the uprights are compressed. The views can be changed by scrolling the uprights. Figure 4.3 shows the procedure of bifocal display in the related work.

### Table Lens

Table lens [82] is a visualization technique to explore very large tabular data. The data items located in the focus are magnified for a normal view while the rest are squeezed into rows of pixels. The information hidden in these pixels can be revealed by selecting and stretching. By this way, it provides much wider scope than conventional table representation. Figure 4.4 gives the display of a table in the work by Rao et al.

### Hyperbolic Representation

Hyperbolic representation technique is applied to visualize large graph structure

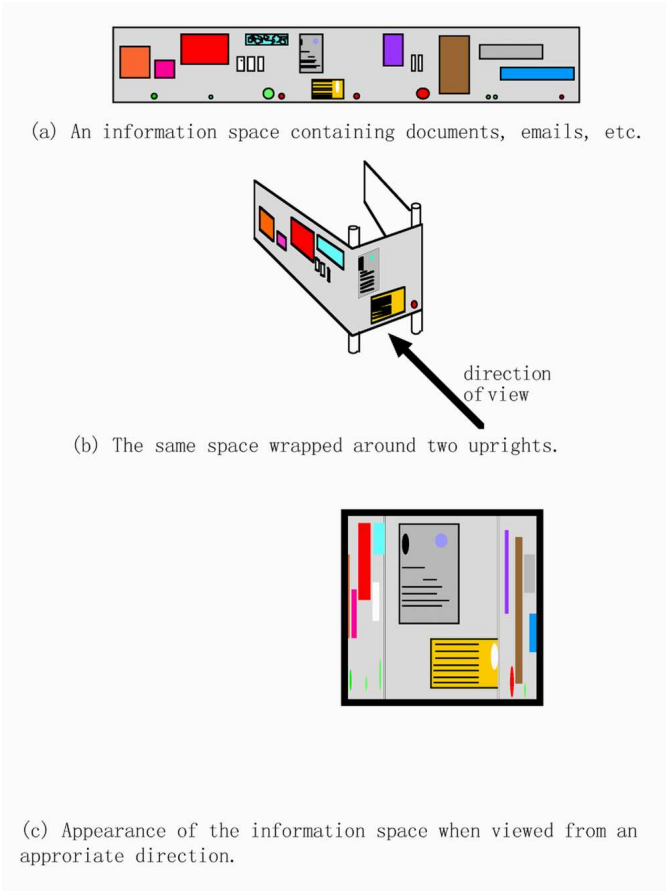



Figure 4.3: Procedure of bifocal display in the work by Spence et al. [61] (Provided by the author).

data, such as trees. It works by first mapping data onto a hyperbolic plane and displaying the plane in a circular space. Thus one data node in the focus can be referred by its context in all directions in the space. The change of focus is controlled through the transformation of the data on the hyperbolic plane. Figure 4.5 gives a hyperbolic display of some Web site information represented as a graph in the work by Munzner [73].

### Remarks

In summary, focus plus context techniques present information both for current attention (focus) and navigation in the environment (context). To achieve this simultaneously in a display screen, distortion is inevitable to suppress data out of focus. This

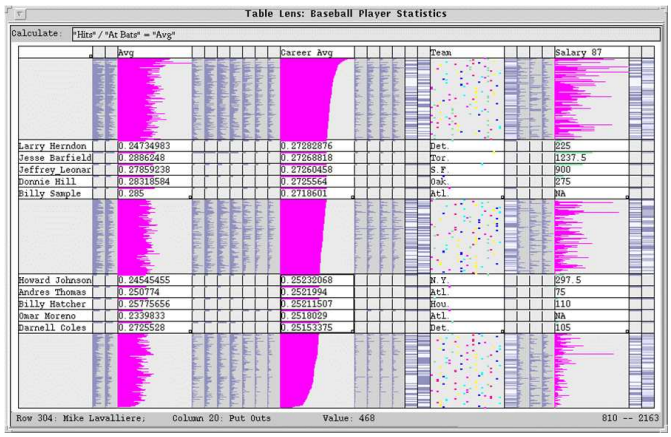



Figure 4.4: Display of large table in the work by Rao et al. [82] (Reproduced with permission of the author).

may not be appropriate to be applied in the situation that accuracy of representation is required. Besides, there will be information clutter caused by the distortion in the display thus making it difficult for readability.

### 4.1.3 Map generalization techniques

The visual readability of the variable-scale displayed map can be improved by generalization techniques. Map generalization is an important procedure in the process of map representation. According to the definition by Tyner [102], it refers to the adjustment of details through selection, simplification and symbolization in order to adapt to the scale of the map. When the map is displayed at a smaller scale, the layout of the objects in the map will become cramped and undistinguishable. Map generalization is designed to solve this problem. The main purpose is to maintain the clarity and readability of the map. It is achieved by a series of spatial or semantic transformations on the map objects. These transformations are generally named as generalization operators. The study of generalization operators has aroused much interest in the literature [72, 68, 19, 78, 6, 83]. However there is no consensus on the definition and classification of generalization operators.

The following gives an overview of the thirteen transformations commonly used in

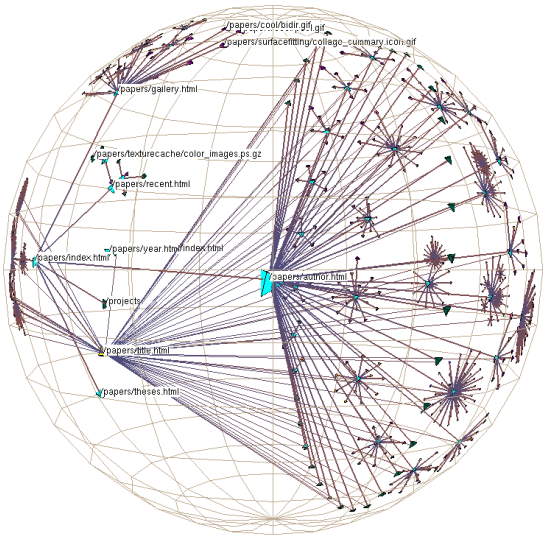



Figure 4.5: Hyperbolic representation of a large graph in the work by Munzner [73] (©1997 IEEE. Reproduced with permission of the author).

map generalization. However the first operation was not included in McMaster and Shea's typology as they regarded it as only a preprocessing step. The other twelve operations can be classified as spatial transformations for the first ten and attribute transformations for the rest. The difference is that the former changes the shape of map objects while the latter changes the topological features.

**Selection** Selection is the most basic operation of map generalization [100]. It is responsible for the choice of map objects to be displayed at a particular scale. The objects that are less important or too small to display are discarded. The judgment for such selection may be largely affected by the purpose of the map. For maps emphasizing on geographic features, the selection is based on the geometric properties of the features. For maps emphasizing on thematic features, the consideration is given less on the physical shapes of the map objects but more on their attributes.

### Spatial transformations

- Simplification: Simplification is also a process of choosing map objects, similar to selection. However, its goal is different. It makes a decision to find an optimal

representation of map objects, for example displaying a polyline with minimum points under certain tolerance error. The simplified map objects usually keep the most salient features. The Douglas-Peucker line simplification algorithm [20] is a representative approach that has been widely employed. The algorithm is an iterative procedure. For each step a point is kept if the distance between it and the line formed by the two end points is larger than a given threshold. This point will divide the original polyline into two pieces and the first step will continue until no points are kept.

- Smoothing: Smoothing is the operation to relocate the vertices of map objects, usually the linear objects, to replace sharp connections with rounded ones. The objective is to improve the legibility of the map as the rounded features provide aesthetically pleasing visual effect. The layout of the map is also simplified since the smoothing has captured the major tendencies of map objects. The difference with simplification operation is that the number of vertices remains the same after smoothing.

- Displacement: Displacement is the technique to rearrange adjacent objects when they are too close for each other due to the decrease of map scale [60]. The relocation of the conflicting objects is performed by applying displacement vectors on their vertices. To find an appropriate position for the vertices is not easy and quite time-consuming. Considerations should be taken that the displacement should not introduce new conflict to surrounding objects. It can be also viewed as a post-processing for some other generalization techniques.

- Refinement: Refinement is the operation to present map objects by representative features. This operation is to solve the problem when there are too many or small features to show clearly. By changing the manner of less important features close to more important ones, the map is depicted concisely. In some sense, this operation has the same function with selection. However this operation cares more

about maintaining the connectivity of the overall representation.

- Exaggeration: Exaggeration is the technique to enlarge the size and shape of particular map objects that otherwise may be too small or narrow to display. It is also helpful to expressly reveal the relationship among map objects, for example the intersection between linear objects. By this transformation, the density will increase at the region holding exaggerated objects thus some other operations, such as selection, may be aroused to improve readability of the map.

- Enhancement: Enhancement has the same objective as exaggeration however it deals with the symbolized map objects. Symbols are employed to differentiate map objects with same geometric shape but different meanings, such as the importance of the ranking of highways. To maintain the clarity of different map objects when displayed in a reduced scale, details are added to highlight the features of these objects.

- Aggregation: Aggregation is the operation to represent a group of adjacent map objects by a singe object thus reducing the number of total objects when the map scale is reduced. The objects in the group may have different features and are joined to form an object with higher-order feature.

- Merging: Merging is the operation similar to aggregation. The difference is that aggregation usually handles point objects while merging handles linear objects.

- Amalgamation: Amalgamation is the operation that combines aggregation and merging. It usually handles two dimensional objects, or areas. This technique is classified into two operations: fusion that connects objects sharing a same boundary and merge that connects objects which are separated. Topological features may be changed caused by amalgamation, for example the changing from point to area, area to line, etc.

- Collapse: Collapse is the operation to convert map objects into the lower-dimensional

representations, such as area to line and line to point. This is because when the map is displayed in a smaller scale some objects may shrink accordingly.

**Attribute transformations**

- Symbolization: Symbolization is the operation to represent classified map objects by graphic symbols to achieve better communication purpose of the map. Depending on the type of map objects, symbols can adopt various visual forms such as shape, color, etc. The symbols may also change their formats when the map scale changes. For example, the boundary of a city may be represented from a polygon to a point when the map scale decreases.

- Classification: Classification is the operation to cluster map objects that share similar features. By this operation, the complexity of a map can be largely relieved. This operation is similar to aggregation, merging and amalgamation. The difference is that this operation emphasizes more on semantic levels.

### 4.1.4 Line smoothing techniques

The map generalization approach used in the thesis is smoothing. The main purpose of line smoothing in map generalization is to improve the readability and the aesthetic appearance of the map. It brings better perceptibility to present line features in rounded format than the connection through sharp corners. Thus it is treated as a cosmetic operation to "iron out" the wrinkles of the linear features.

Line smoothing issues have been extensively studied in the literature work. Some well-known algorithms may include point averaging, tolerancing and curve fitting [69].

Point averaging method computes the new location of each point from the average of its neighboring points. The influence of the neighboring points can be adjusted by some weighting function, such as Gaussian function [2]. Tolerancing method smooths lines by rolling a circle of diameter $\epsilon$ along the line [79]. The indentations that are not covered or touched by the circle are eliminated and replaced by a line. Curve fitting method

approximates lines by the convolution with some mathematical functions. Although this method is more complex than the previous ones, it provides much flexibility to control the shape of the target curves. Representative functions are Splines [88, 89] and Bézier curves.

In the context of map generalization, the employment of line generalization techniques is usually connected with some controllable factors, such as topological constraints. As the map can be treated as a collection of polygonal contours, the processing of each object has to respect the consistency of its environment. There are a few work catered for this requirement. Mustafa et al. [75] gave an algorithm for map simplification with the constraint of non-intersection. They modeled a given map by a set of piecewise connected chains. To prevent intersection among these chains during simplification, they built Voronoi regions for each chain. The observation is that if all the chains are simplified within their respective Voronoi regions, they will not intersect with other chains. Thus the Voronoi regions are the guaranteed districts for the constraints. The similar idea can be traced to the work on mesh approximation by simplification envelopes [18] and line simplification by safe set [64].

To handle the combination of various generalization constraints, Berg et al. [8] gave an algorithm on map simplification. The main contribution is that they enforced the constraints by maintaining the relationship between the curve under simplification and the rest in the map. If the relationship is consistent, the constraints are satisfied. However they have not given implementation on real map data. There are some other work based on optimization techniques proposed to provide an overall solution [10, 38]. The main approaches include force models such as springs [10] and snakes [14, 12, 13, 95], simulated annealing [107], beams [3] and Least Squares Adjustment [36, 92].

There is a common drawback for most line smoothing techniques that the smoothed contours may shrink in the long run. Some methods are proposed to correct this problem by intentionally compensating the region under shrinkage. Lowe [58] gave an algorithm based on standard Gaussian smoothing. The improvement is that the shrinkage caused

by the smoothing is calculated and counteracted by inflating the curve to the same amount. Similar work were given by Vollmer et al. [106], Kuprat et al. [48] and Hahmann et al. [34]. The main idea is to relocate the vertices of the smoothed curves to correct the area shrinkage. However these work can only handle single curve or closed curve. Another methods directly embedded area preserving in the process of curve deformation, such as adding low-pass filter effect on Gaussian smoothing [97, 77] , mean curvature flow [40, 26] and Euclidean geometric heat flow [85]. The drawbacks are the operation is slow due to the locality property for the first one and the requirement of a complex PDE solver for the latter two.

### 4.1.5 Constraint-based map generalization

Map generalization is not an isolated event for each map object, some measurements should be taken to insure the overall correctness after generalization [108]. Generally, these measurements or constraints are defined as a set of designed specifications that controls the process to obtain the solution of map generalization [109]. According to some research work [80, 109], the generalization constraints are divided into four categories as follows:

- Graphical constraints: This class of constraints controls the perceptibility for viewers through setting graphic limits. It decides both the individual and proximity properties, for example the setting of the minimum size of map objects and the minimum distance among map objects.

- Topological constraints: This class of constraints controls the preservation of the topological relationship among map objects, such as intersection and connectivity. It is also responsible for the topological consistency of individual objects, for example the prevention of self-intersection of polygonal lines during generalization.

- Structural constraints: There are two components included in this class of constraints which are spatial structural constraints and semantic structural con-

straints. The former emphasizes on the preservation of geometrical properties of map objects, such as their shapes and the alignments. The latter emphasizes on the logical aspects, for example, the generalization of the coastline should not make the city into the sea.

- Gestalt constraints: This class of constraints is related to the visual aspects of the generalization. It aims to achieve the aesthetic impression of the overall distribution of the map objects such as the visual balance of the map. In contrast to the other constraints, gestalt constraints are largely dependent on the perceptibility of human viewer, thus making it difficult to formulize into a sequence of operations. It is also recommended that this evaluation should be performed in the final step.

## 4.2 Proposed method

### 4.2.1 Motivation

With the advances of satellite imaging technology, the large amount of high resolution geospatial data-sets becomes available. For example, the total size of the TIGER data-set [104], provided from the US Census Burean that gives the geographic description of the whole US by the vector-based representation, is about 40GB in uncompressed format and 3.5GB in the zipped form that consists of more than 3000 archive files. In contrast to this data explosion, the size of the display devices is becoming smaller due to the growing popularity of personal computing facilities such as portable computers and mobile phones. This leads to a challenge: how to render large data-set in a relatively small display. This is known as the "maximize the usage of screen real estate problem" in visualization.

On the other hand, the map with more information may disturb viewer to grasp the most relevant knowledge. For example, in driving navigation, the sinuosities along the route, although reflecting the actual roads condition, may cause information clutter to the viewer.

To address the problem due to the small window size, variable-scale display is a natural approach [35]. This is motivated by the fact that in such applications, it is required to provide both a detailed information denoting the users' location and a context information giving the guiding direction. This is especially important in the applications of personal navigation like Location Based Services as the successfulness of way-finding is largely dependent on the references.

As for the second problem on information cluttering, simplification techniques, including distortion and abstraction, are designed to improve the readability of the map [59, 70]. For most work in this field, the main objective is to remove less important details while salient features are retained. One drawback of such simplification is that it usually lacks aesthetic consideration as sharp turns may appear. Thus it is expected that the map is represented by a set of succinct and smooth curves with topological fidelity.

### 4.2.2 A general approach

We take the following steps to generalize a map and render it in a small display window.

- Step 1: Objects outside of the ROI that have no relationship with objects in the ROI are discarded.
- Step 2: A fisheye lens transformation is applied on the retained objects from previous step.
- Step 3: The fisheye transformed map is smoothed.

In this thesis, instead of working on various objects representation, we explore polygonal line or polyline, which represents road or coastline. Other objects like labels, county boundaries, buildings, etc are not considered.

The focus of our work is on step 3. Given a collection of polylines $\mathcal{L}$, we intend to find smoothed lines that achieve optimality while satisfying topographic constraints. The optimality lies in two aspects: to smooth the lines, and to reduce the derivation

from the original. Formally, the optimization process is measured by a score function subjected to certain constraint as follows:

$$min : E = \sum_{l \subset \mathcal{L}} (E_s(l) + E_d(l)) \tag{4.1}$$

where $E_s(l)$ is the sum of curvature along a polyline $l$ and $E_d(l)$ is the derivation of smoothed $l$ from its original.

The curvature of a smooth curve can be defined as the inverse of the radius of the inscribed circle at each point along the curve. For the definition of curvature on polyline by the connection of discrete data points, one possible way to estimate the curvature of a point is to compute the inverse of the radius of the circle passing through the point and its two neighbours. As shown in Figure 4.6 (a), the curvature of a point $P_i$ in the polyline can be given as $k(P_i) = 1/r$ where $r$ is the radius of the circle that passes through 3 consecutive points $P_{i-1}$, $P_i$ and $P_{i+1}$. In the implementation, we simply use the length of the polyline to represent the sum of the curvature along the polyline as intuitively the length can reflect the bending extent of the polyline.

The other item $E_d(l)$ can be measured using Hausdorff distance. However, due to the large number of points involved in the computation, we approximate $E_d(l)$ by the area exchanged between two polylines. For example, in Figure 4.6 (b), the derivation between the polyline and the smooth curve represented by dotted line is measured by the area difference under these two.

The topological constraint is:

- $\mathcal{C}$: the areas enclosed by polylines are preserved.

Note that $\mathcal{C}$ implies that no intersection or self-intersection among the polylines are created or removed.

In the following, we first give a brief description of step 1 and step 2. In Section 4.3, we give a detailed description of step 3.

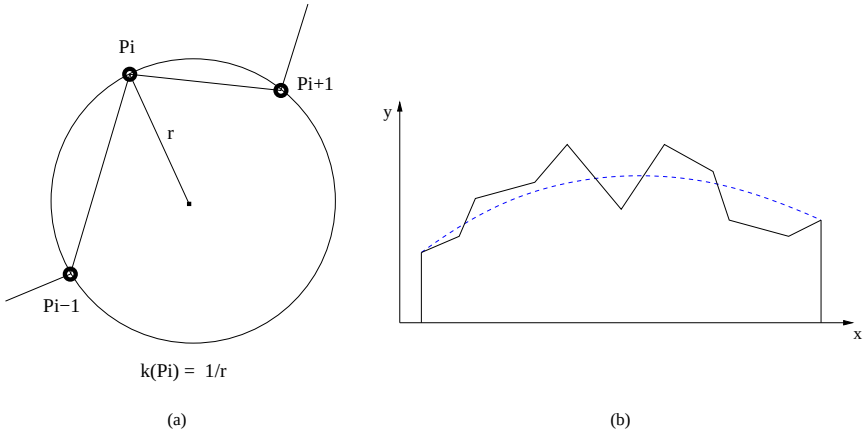



Figure 4.6: (a): Curvature of a point in a polyline. (b):Derivation between a polyline and the smooth curve.

### 4.2.3 Objects filtering and fisheye transformation (Step 1 and 2)

**Filtering non-related objects**

In this step, the map objects having relationship with ROI are retained and the rest are discarded. In the implementation, we denote the ROI as a circular region. If an object has overlapped with the ROI, we consider it relevant to current navigation task. For example, in Figure 4.13 (a) we give an example of a route map marked with a ROI.

**Fisheye view mapping function**

There are two main variations of fisheye view mapping: Cartesian and Polar transformation. Cartesian transformation is applied in rectangular coordinates while Polar transformation is applied in polar coordinates. The difference is that the former transforms data independently on X and Y directions while the latter transforms data in the radial direction originating from the focus.

We employ normalized polar fisheye transformation as the distortion technique. Under this variation, a point $p(x, y)$ in rectangular coordinates is represented by its normal coordinates $(r_{norm}, \theta)$ with the focus $p_f(x_f, y_f)$ as the origin where $r_{norm} = ||p - p_f||$ and $\theta = atan(\frac{y-y_f}{x-x_f})$. The relationship between a point's polar coordinates $(r_{norm}, \theta)$ and its fisheye coordinates $(r_{feye}, \theta)$ is given according to the following distortion function:

$$r_{feye} = r_{max} * G(z, d); \tag{4.2}$$

where $G(z, d) = \frac{(d+1)*z}{d*z+1}$ and $z = \frac{r_{norm}}{r_{max}}$. $d$ is the distortion factor that controls the intensity of the distortion and can be controlled through input from viewer. $r_{max}$ is the maximum bounding value of radius $r_{norm}$ along the direction of $\theta$. $\theta$ remains constant during the transformation.

## 4.3 Line smoothing (Step 3)

### 4.3.1 Main idea

Finding the global optimal solution of E.q. 4.1 seems to be difficult. Instead, we gave a heuristic that iteratively performed local smoothing. Each "local smoothing" essentially finds two Bézier curves that satisfy the constraint $\mathcal{C}$.

The set of polylines is smoothed one by one. For each polyline, it is randomly divided into sub-polylines. The sub-polylines are divided in a way that the smoothing problem on the set of polylines can be reduced to a sequence of simplified instances. To ensure that there is a smooth transition from a sub-polyline to the next one, during the smoothing of a sub-polyline, we include an additional constraints on the tangent of the starting and ending points of the sub-polylines.

A complication appears to process open curves. Note that there is no "enclosed area" under an open curve, and it is not clear on how to impose the area preserving constraint on it. This will be discussed in Section 4.3.4.

### 4.3.2 Algorithm flow

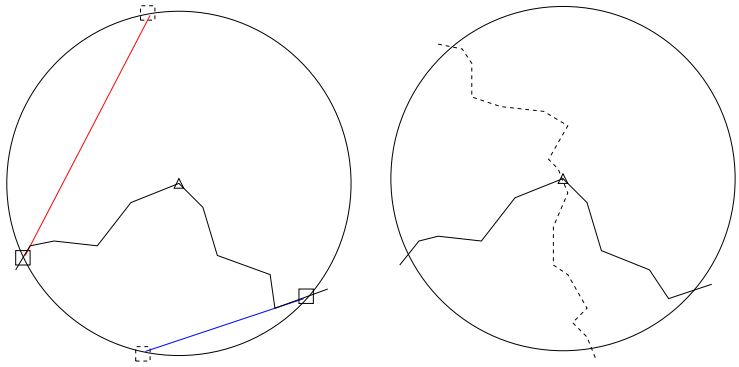
The overall flow of the algorithm is as follow:

1. Pick a non-smoothed polyline $l$ from the collection $\mathcal{L}$.

(a) Randomly pick a point $p_c$ from $l$. Noted that the $p_c$ is either the node of the $l$ or the intersection point resided in the $l$. Find a circle with $p_c$ as the center that the set of sub-polylines $S$ from $\mathcal{L}$ covered by the boundary of the circle should all pass through $p_c$. The radius of the circle is obtained to the largest extent. The relationship between $S$ and the circle belongs to the following two cases:

- Case 1: The center of the circle $p_c$ is the node point of the polyline. There is only one sub-polyline in $S$.
- Case 2: The center of the circle $p_c$ is the intersection point of polylines. There are more than one sub-polylines in $S$ that pass through the center $p_c$. In our current implementation, we only consider two sub-polylines. It can be easily extended to more sub-polylines passing through the intersection point.

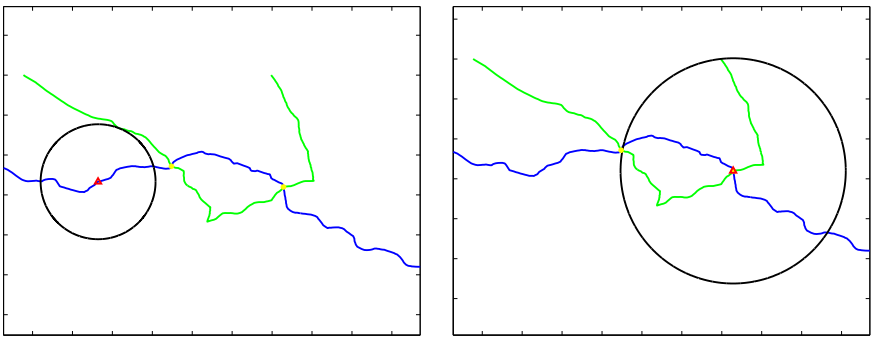
These two cases are illustrated in Figure 4.7. Figure 4.8 gives a snapshot of the moving circle along two polylines. The radius of the black moving circle is obtained at the largest extent.



(a) Case 1 (b) Case 2

Figure 4.7: The relationship between the line segments and the circle. The center of the circle is marked by a triangle symbol.

(b) For each sub-polyline $s_i \subset S$, perform a sub-problem to find the smoothed curve.



(a) (b)

Figure 4.8: The snapshot of the moving circle along two polylines. The yellow dots represent the intersection points of the two polylines. The red triangle represents the center of the moving circle.

2. Process next polyline which is not smoothed.

### 4.3.3 Local smoothing in the sub-problem

The sub-problem is to find the smoothed curve(s) which minimize E.q. 4.1 under the constraint $\mathcal{C}$ for the two cases given above. We employ area-preserving Bézier curve fitting for the smoothing process. Generally, a Bézier curve can be determined by a number of control points. It is very convenient to adjust the shape of the curve by moving these control points. The advantage is that we can easily regulate the area under the curve.

Figure 4.7 (a) shows the smoothing process for case 1. The process consists of the following three steps.

1. **Locating the control points.**

   We use four control points $P_1(A_x, A_y)$, $P_2(B_x, B_y)$, $P_3(C_x, C_y)$, $P_4(D_x, D_y)$ to determine a Bézier curve. Parametrically, the Bézier curve is represented as follow:

$$
\begin{aligned}
x(u) &= A_x(1-u)^3 + 3B_x(1-u)^2u + 3C_x(1-u)u^2 + D_xu^3 \\
y(u) &= A_y(1-u)^3 + 3B_y(1-u)^2u + 3C_y(1-u)u^2 + D_yu^3 \\
u &\subset [0,1]
\end{aligned}
\tag{4.3}
$$

The coefficients in the equation represent the coordinates of the control points. Points $P_1$ and $P_4$ are the two end points of the Bézier curve and are obtained as the intersection points between the polyline and the circle. Points $P_2$ and $P_3$ are the two intermediate points. The adjustment of these two points will affect the overall shape of the obtained Bézier curve. These two points are obtained by extending the line segment from the polyline that intercepts the circle until getting another intersection with the circle. These extended line segments are marked as red and blue in Figure 4.7 (a). The intersection points are represented as a square with solid line and dashed line separately. Noted that the dashed-line square is obtained from the extension. For the solid-line squares, they are fixed as control points and decide the tangent vector at the two ends. The other two control points are chosen from the red and blue lines by some sampling interval.

2. **Obtaining Bézier curve under constraints.**

   The obtained Bézier curve through fitting on the control points located in the first step should not cross over the circle and the new partition made on the area of the circle should be the same as that divided by the original line segments. The first requirement has insured that no imported intersection points occur and the second one is for the area preserving constraint. In our implementation, we consider it acceptable if a enclosed area is not altered substantially. Thus we set a threshold $\gamma$ as the limitation for the percentage of area difference between a smoothed polyline and its original.

3. **Finding the optimal solution.**

   There may be more than one Bézier curve obtained in step 2. These curves are the candidates for choosing the optimal solution judged by E.q. 4.1. Considering the relatively small parameter space, we use exhaustive search to find the optimal Bézier curve.

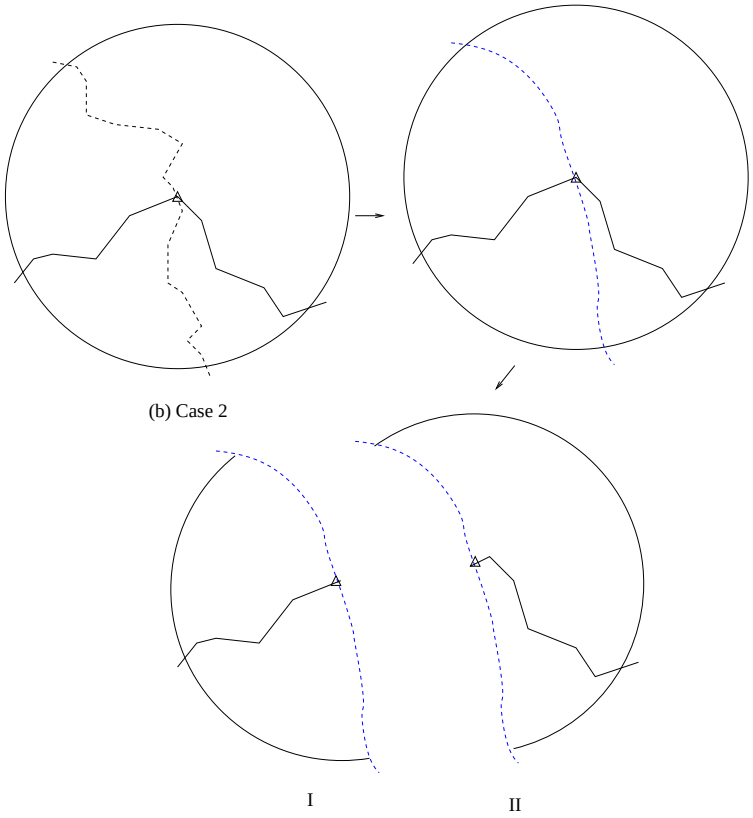



Figure 4.9: The smoothing on Case 2. It is referred to as Case (b) in Figure 4.7.

For smoothing on case 2 that there are two sub-polylines in the safe region, we give the following algorithm.

There are two steps for the algorithm. We first process one sub-polyline with the whole circle as the boundary. The problem can be reduced to case 1. As shown in Figure 4.9, the blue dotted line represents the Bézier curve fitting for the dotted polyline in Case 2. The following step is to process the two parts of the other sub-polyline. The smoothing of each part can also be reduced to case 1. To handle multiple polylines, the

center needs to be fixed after the first step.

### 4.3.4 Area-preserving on open curves

To handle the area-preserving constraint on open curves, we adopt the idea of "diff-area model" in the work by Bose et al. [11]. This model gives a solution to simplify a polygonal path with the objective that the area above (or below) the path changed by the approximation is preserved. However their work is constrained to handle x-monotone polygonal path. Here we give a simple overview of the model.

*Problem definition.* Suppose $P$ and $Q$ are two $x$-monotone polygonal paths where $Q$ is the approximating one of $P$. Let $\Omega_A(Q)$ be the area above $P$ and below $Q$, and $\Omega_B(Q)$ be the vice versa.

*Diff-area model.* To measure the quality of the approximation by $Q$, diff-area model suggests that the cost function is $|\Omega_A(Q) - \Omega_B(Q)|$ which is the area exchange above (or below) $P$ and $Q$. This model implies that the area is "exchangeable", that is, area loss at one location can be redeemed from another location.

In our algorithm, each polyline has been divided into a set of sub-polylines such that the smoothing problem is easy to handle for each sub-polyline. This division has also created the flexibility to apply diff-area model in each sub-polyline. As each sub-polyline is smoothed in a circle locally found, applying diff-area model is identical to preserve area of the circle partitioned by the sub-polyline. The overall area-preserving constraint is satisfied while the local smoothing process is performed iteratively.

## 4.4 Implementation and experiments

We implemented our algorithm on a synthetic data-set and a real map data-set.

Figure 4.10 (a) gives the full view of the synthetic data-set with 9 polylines in ROI are highlighted. The ROI is simply denoted by a black circle. All the polylines outside the ROI or the black circle are regarded as non-related to current navigation interest

and filtered. The retained 9 polylines are depicted by different colors in Figure 4.10 (b). For the purpose of easy illustration, we mark each route by a sequence number at one end of the route.

Figure 4.11 gives the experimental results of our algorithm with different parameters. Figure 4.11 (a) and Figure 4.11 (b) give the fisheye transformation with different focus of interest. The distortion factor $d$ is 3 and the focus is marked as a red dot. Figure 4.11 (c) and Figure 4.11 (d) give the smoothing results on Figure 4.11 (a) and Figure 4.11 (b). The threshold $\gamma$ for measuring the area difference is set as 0.01. Figure 4.11 (e) and Figure 4.11 (f) have the same parameters as Figure 4.11 (c) and Figure 4.11 (d) except that $\gamma$ is set as 0.3.

Note that the routes after smoothing have much less sinuosities comparing to their original counterparts. Meanwhile, the area enclosed by the routes are not changed drastically. Although our algorithm performs smoothing locally, the global smoothness for the routes are achieved. Taking polyline 8 in Figure 4.11 (e) as an example, the 8 segments partitioned by the 7 intersection points not only stay smooth separately but also keep continuous on the conjunction points. Increasing $\gamma$ may achieves better smoothing effect while the enclosed area is altered significantly.

Figure 4.12 gives the overall score function $E$ for polyline 1 in Figure 4.11 (c). As we consider the area difference negligible, this curve reflects the curvature variation at each iteration. It shows that the solution provided by our algorithm always improves the optimality of the route.

The real map data-set is extracted from a simple representation of the major roads in the state of Connecticut, US. It depicts the highway network in the state at 1:250,000 scale. Figure 4.13 gives the results on our algorithm.

## 4.5 Remarks

In this work, we design a map generalization algorithm that handles visualization of vector-based map on devices with small display window. The algorithm firstly filters out

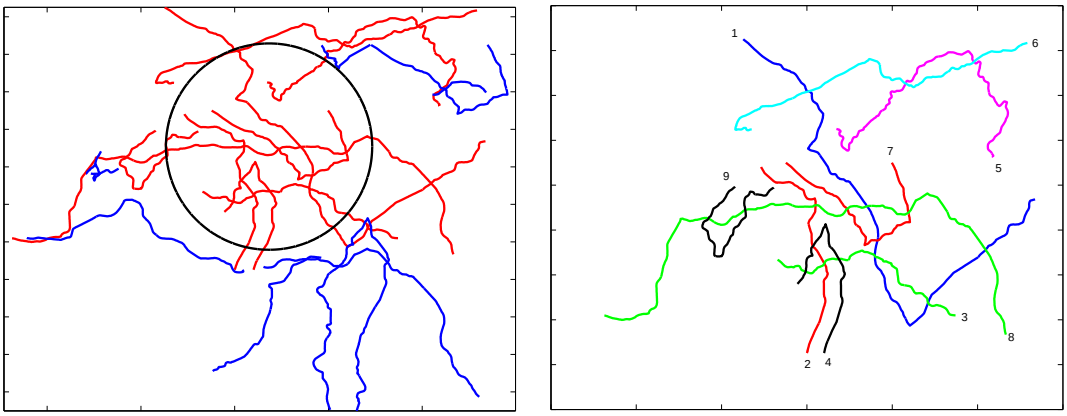
non-related information for current navigation task. The following steps are applying fisheye lens to exaggerate objects in the ROI and smoothing the lines to remove clutter caused by the distortion. Our main focus is on the line smoothing part. We presented an algorithm that formulates the smoothing process as an optimization problem which minimizes the overall curvature while preserving the enclosed area by the lines. We gave a heuristic method to find the optimal solution by dividing the problem into the combination of a set of sub-problems. For each sub-problem, our objective is to find at most two Bézier curves given at most two polygonal lines which intersect at most once in a circular region, meanwhile the partitioned area of the region by the two lines are preserved. Experimental study demonstrates that our algorithm achieves the approximation of global optimality for the generalized map.

There are three aspects to extend current work.

1. In each sub-problem, we determine the Bézier fitting for each polygonal segment by a small number of parameters. Thus the whole polyline could be represented by a compact format. This could be used in remote visualization by transmitting only the compact format of a map thus improving the efficiency. However, as the sub-problems are independent for each other, consideration should be given on how to reconstruct the correct final presentation upon receiving the transmitted parameters.

2. Currently, we only consider the geometric features of a vector-based map. Generalization may become complex when the other features are considered, for example, the labeling of map objects. The positioning of the label will be a challenging issue when the corresponding object is relocated.

3. The efficiency of the algorithm can be improved if the process of choosing segments to smooth is controlled. There are a few crucial points on each polyline that dominant the major bending energy. The detection of these points is critical to accelerate the smoothing iteration.




(a) (b)

Figure 4.10: The synthetic data with 9 polylines in ROI. The 9 polylines are depicted as red color while the rest are in blue. The ROI is denoted as the black circle.

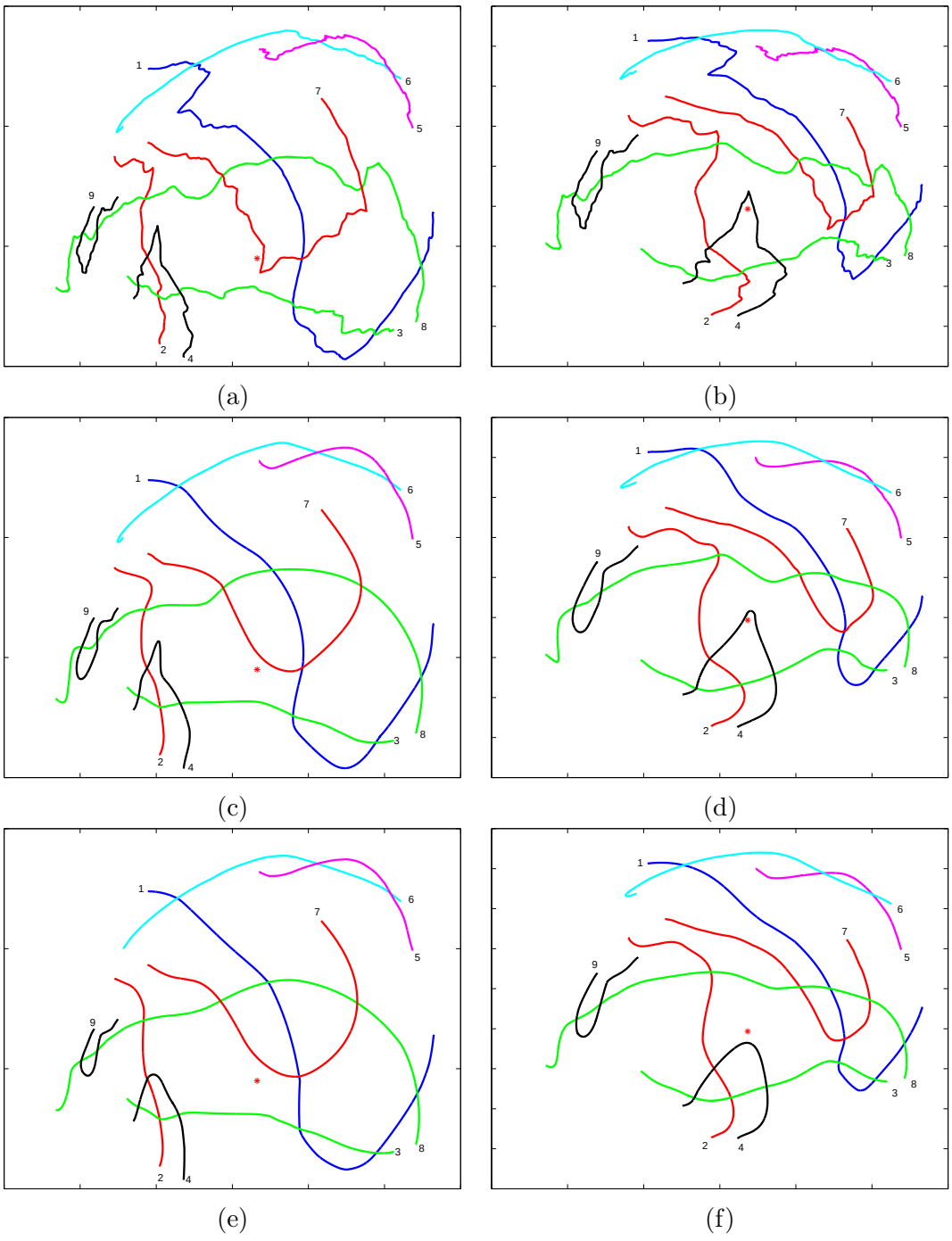



Figure 4.11: Fisheye view transformation plus line smoothing on 9 polylines. Distortion factor $d = 3$. Area difference threshold $\gamma = 0.01$ for (c) and (d). $\gamma = 0.3$ for (e) and (f).

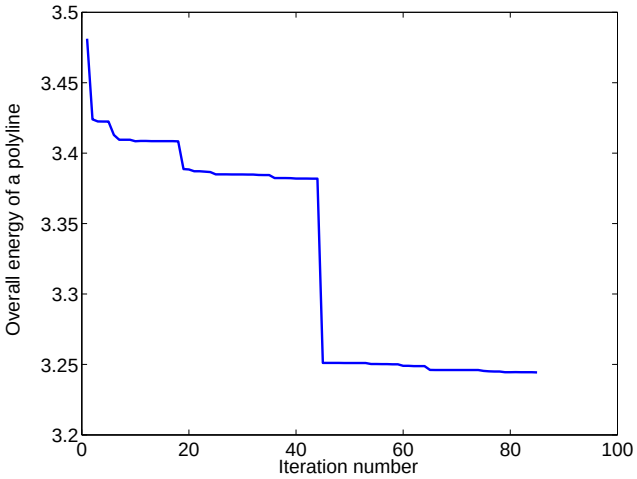



Figure 4.12: Overall energy of a polyline 1 in Figure 4.11 (c) versus the iteration number.

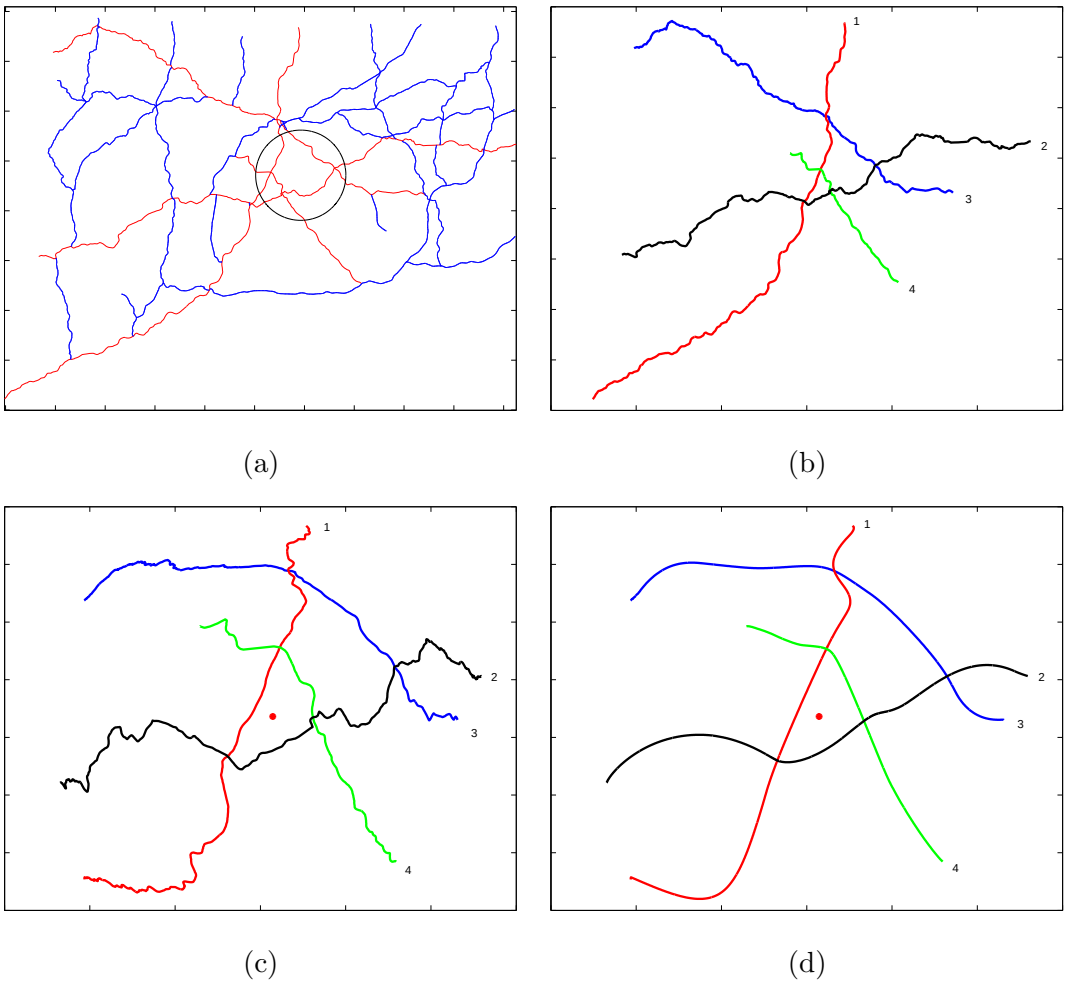



Figure 4.13: The route map data-set is extracted from a simple representation of the major roads in the state of Connecticut, US. It depicts the highway network in the state at 1:250,000 scale. The black circle in (a) indicates the ROI with 4 routes depicted as red color. Fisheye view distortion factor $d = 3$ for (c). Area difference threshold $\gamma = 0.01$ for (d).

# Chapter 5

# Conclusions

The thesis discusses some issues on visualization with limited resources at the viewer's side. In this work we consider two forms of the resources: the computing power and the size of display window. We adopt region of interest (ROI) techniques as the potential solution to maximize the resources usage. ROI approach has two advantages: 1) it intentionally allocates more resources to the interesting region; 2) it leads the viewer's attention to the interesting region. In order to improve the information readability, smooth transition is advocated to alleviate the discontinuity between object from high to low level of interest. We study the variations of ROI techniques in the context of two applications: remote volume visualization and vector-map visualization.

The first part of the thesis studies the remote visualization of volume data where the client has access to low computing resources. We adopt foveation approach in which volume data are represented by multiple levels of resolution with the highest in the ROI. One technical issue of this approach is on how to efficiently render the foveated volume. We give an algorithm that renders the foveated volume directly in the wavelet domain. The rendering time only depends on the relevant wavelet coefficients of the foveated volume. Another issue is on how to effectively visualize a foveated volume as the overall resolution is reduced. We give methods to highlight objects in ROI thus the overall quality is not affected drastically in ROI.

The second part of the thesis studies the visualization of vector-based map in small display window. To cater for the requirement of presenting large data in the limited space, we design a map generalization algorithm that shows relevant navigation information in a variable-scale fashion. We adopt fisheye transformation to display the objects in the ROI in a larger scale. This operation may inevitably cause information clutter at the peripheral due to the distortion. To solve this problem, we present a line smoothing algorithm. The smoothing process is an iteration of localized smoothing procedure that satisfies topological constraints.

# Appendix

**List of publications:**

1. Hang Yu and Ee-Chien Chang, Distributed Multivariate Regression Based on Influential Observations, 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, page 679-684.

2. Hang Yu, Vu Thanh Nguyen and Ee-Chien Chang, Rotation of foveated image in the wavelet domain, IEEE International Conference on Image Processing, 2004.

3. Hang Yu, Ee-Chien Chang, Zhiyong Huang and Zhijian Zheng, Fast Rendering of Foveated Volumes in Wavelet-based Representation, 13th Pacific Conference on Computer Graphics and Applications, 2005. (published in The Visual Computer (TVC)).

# Bibliography


[1] The VolPack volume rendering library. http://graphics.stanford.edu/software/volpack/, 1995.

[2] J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda. Uniqueness of the gaussian kernel for scale-space filtering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(1):26–33, 1986.

[3] M. Bader. *Energy Minimization Methods for Feature Displacement in Map Generalization.* PhD thesis, Univ. Zürich, Mathematisch-naturwissenschaftlichen Fak., 2001.

[4] A. Basu, I. Cheng, and Y. Pan. Foveated online 3D visualization. In *ICPR '02: Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02) Volume 3*, page 30944, Washington, DC, USA, 2002. IEEE Computer Society.

[5] A. Basu and K. J. Wiebe. Video conferencing using spatially varying sensing with multiple and moving fovea. *IEEE Trans. on Systems, Man and Cybernetics*, 28(2):137–148, 1998.

[6] M. K. Beard and W. A. Mackaness. Generalization operations and supporting structures. In *Proceedings, Tenth International Symposium on Computer-Assisted Cartography (Auto-Carto 10)*, pages 29–45, 1991.

[7] B. B. Bederson, J. D. Hollan, K. Perlin, J. Meyer, D. Bacon, and G. W. Furnas. Pad++: A zoomable graphical sketchpad for exploring alternate interface physics. *Journal of Visual Languages and Computing*, 7(1):3–32, 1996.

[8] M. D. Berg, M. V. Kreveld, and S. Schirra. A new approach to subdivision simplification. In *Twelfth International Symposium on Computer- Assisted Cartography*, volume 4, pages 79–88, Charlotte, North Carolina, 1995.

[9] J. F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. In *SIGGRAPH '82: Proceedings of the 9th annual conference on Computer graphics and interactive techniques*, pages 21–29, New York, NY, USA, 1982. ACM Press.

[10] J. Bobrich. *Ein neuer Ansatz zur Kartographischen Verdrängung auf der Grundlage eines mechanischen Federmodells.* PhD thesis, 1996.

[11] P. Bose, O. Cheong, S. Cabello, J. Gudmundsson, M. V. Kreveld, and B. Speckmann. Area-preserving approximations of polygonal paths. *Journal of Discrete Algorithms*, 4:554–566, 2006.

[12] D. Burghardt. Glättung mit snakes. *Festschrift zum 65. Geburtstag von Prof. Dr. Ing. habil. Siegfried Meier, Technische Universität Dresden*, 2002.

[13] D. Burghardt. Controlled line smoothing by snakes. *GeoInformatica*, 9(3):237–252, 2005.

[14] D. Burghardt and S. Meier. Cartographic displacement using the snakes concept. *Frstner W. and Plmer L. (eds.), Semantic Modelling for the Acquisition of Topographic Information from Images and Maps*, pages 59–71, 1997.

[15] E. C. Chang. *Foveation Techniques and Scheduling Issues in Thinwire Visualization.* PhD thesis, Computer Science, Courant Institute, New York University, May 1998.

[16] E. C. Chang, S. Mallat, and C. Yap. Wavelet foveation. *Journal of Applied and Computational Harmonic Analysis*, pages 312–336, 2000.

[17] H. E. Cline, W. E. Lorenson, S. Ludke, C. R. Crawford, and B. C. Teeter. Two algorithms for the 3D reconstruction of tomograms. *Medical physics*, 15(3):320–327, 1988.

[18] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. *Computer Graphics*, 30(Annual Conference Series):119–128, 1996.

[19] G. Dettori and E. Puppo. Designing a library to support model-oriented generalization. In *ACM-GIS*, pages 34–39, 1998.

[20] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[21] S. Dunne, S. Napel, and B. Rutt. Fast reprojection of volume data. In *Proceedings Of The First Conference on Visualisation And Biomedical Computing*, pages 11–18, 1990.

[22] D. Ebert and R. Parent. Rendering and animation of gaseous phenomena by combining fast volume and scanline A-buffer techniques. *ACM SIGGRAPH Computer Graphics*, 24(4):357–363, 1990.

[23] D. S. Ebert, R. Yagel, J. Scott, and Y. Kurzion. Volume rendering methods for computational fluid dynamics visualization. In *IEEE Visualization*, pages 232–239, 1994.

[24] A. B. Ekoule, F. C. Peyrin, and C. L. Odet. A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Trans. Graph.*, 10(2):182–199, 1991.

[25] A. Eleftheriadis and A. Jacquin. Automatic face location detection and tracking for model-assisted coding of video teleconferencing sequences at low bitrates. *Signal Processing: Image Communication*, 7(3):231–248, 1995.

[26] J. Escher and G. Simonett. The volume preserving mean curvature flow near spheres. In *Proceedings of the American Mathematical Society*, volume 126, pages 2789–2796, 1998.

[27] F. C. A. Fernandes, R. L. C. V. Spaendonck, and C. S. Burrus. A new framework for complex wavelet transforms. *IEEE Transactions on signal processing*, 51:1825–1837, 2003.

[28] R. A. Fisher and H. M. Tong. A full-field-of-view dome visual display for tactical combat training. *Proc. Image Conference IV*, pages 23–26, June 1987.

[29] N. Fout, H. Akiba, K. L. Ma, A. Lefohn, and J. Kniss. High quality rendering of compressed volume data formats. In *EuroVis 2005*, 2005.

[30] B. Fröhlich, S. Barrass, B. Zehner, J. Plate, and M. Göbel. Exploring geo-scientific data in virtual environments. In *IEEE Visualization*, pages 169–173, 1999.

[31] G. W. Furnas. Generalized fisheye views. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23, New York, NY, USA, 1986. ACM Press.

[32] A. Gröpl, T. Günther, J. Hesser, J. Kröll, R. Männer, C. Poliwoda, and C. Reinhart. A Volume Rendering System for Medicine. In *Proc. Informationssysteme im Unternehmen Krankenhaus*, number 1-3, Heidelberg, 1995.

[33] M. H. Gross, L. Lippert, R. Dittrich, and S. Häring. Two methods for wavelet-based volume rendering. *Computers and Graphics*, 21(2):237–252, 1997.

[34] S. Hahmann, B. Sauvage, and G. P. Bonneau. Area preserving deformation of multiresolution curves. *Computer Aided Geometric Design*, 22(4):349–367, 2005.

[35] L. Harrie, L. T. Sarjakoski, and L. Lehto. A variable-scale map for small-display cartography. In *Proc. Symposium on GeoSpatial Theory, Processing, and Applications*, pages 8–12, 2002.

[36] L. Harrie and T. Sarjakoski. Generalization of vector data sets by simultaneous least squares adjustment. *International Archives of Photogrammetry and Remote Sensing*, XXXIII(A4):340–347, 2000.

[37] G. T. Herman and H. K. Liu. Three-dimensional display of human organs from computed tomograms. In *Computer Graphics and Image Processing*, volume 9, pages 1–21, 1979.

[38] P. Hójholt. Solving local and global space conflicts in map generalization using a finite element method adapted from structural mechanics. In *Proceedings 8th International Symposium on Spatial Data Handling*, pages 679–689, 1998.

[39] K. Hornbaek and E. Frr. Reading of electronic documents: the usability of linear, fisheye, and overview+detail interfaces. In *CHI*, pages 293–300, 2001.

[40] G. Huisken. The volume preserving mean curvature flow. *J. Reine Angew. Math.*, 382:35–48, 1987.

[41] I. Ihm and R. K. Lee. On enhancing the speed of splatting with indexing. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 69, Washington, DC, USA, 1995. IEEE Computer Society.

[42] J. T. Kajiya and B. P. V. Herzen. Ray tracing volume densities. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 165–174, New York, NY, USA, 1984. ACM Press.

[43] A. C. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging*. New York: IEEE Press, 1988.

[44] A. Kaufman. *Introduction to volume visualization.* IEEE Computer Society Press, 1991.

[45] T. Alan Keahey. The generalized detail-in-context problem. In *Proceedings IEEE Symposium on Information Visualization 1998*, pages 44–51, 1998.

[46] D. A. Keim and H. P. Kriegel. Visualization techniques for mining large databases: A comparison. *IEEE Trans. Knowl. Data Eng.*, 8(6):923–938, 1996.

[47] H. Koike. Generalized fractal views. In *Proc. of Advanced Visual Interfaces*, 1992.

[48] A. Kuprat, A. Khamayseh, D. George, and L. Larkey. Volume conserving smoothing for piecewise linear curves, surfaces, and triple lines. *Journal of Computational Physics*, 172(1):99–118, 2001.

[49] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 451–458, New York, NY, USA, 1994. ACM Press.

[50] D. Laur and P. Hanrahan. Hierarchical splatting: a progressive refinement algorithm for volume rendering. *SIGGRAPH Comput. Graph.*, 25(4):285–288, 1991.

[51] R. K. Lee and I. Ihm. On enhancing the speed of splatting using both object- and image-space coherence. *Graphical models*, 62(4):263–282, 2000.

[52] S. Lee and A.C. Bovik. Fast algorithms for foveated video processing. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(2), 2003.

[53] M. Levoy. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, 8(3):29–37, 1988.

[54] M. Levoy. Efficient ray tracing of volume data. *ACM Trans. Graph.*, 9(3):245–261, 1990.

[55] M. Levoy. Volume rendering, a hybrid ray tracer for rendering polygon and volume data. *IEEE Computer Graphics and Applications*, 10(2):33–40, 1990.

[56] M. Levoy and R. Whitaker. Gaze-directed volume rendering. In *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 217–223, New York, NY, USA, 1990. ACM Press.

[57] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM Press.

[58] D. G. Lowe. Organization of smooth image curves at multiple scales. Technical report, 1989.

[59] A. M. MacEachren. *How maps work*. Guilford Publications, 1995.

[60] W. Mackaness. An algorithm for conflict identification and feature displacement in automated map generalization. *Cartography and Geographic Information Science*, 4(21):219–232, 1994.

[61] J. D. Mackinlay, G. G. Robertson, and S. K. Card. The perspective wall: detail and context smoothly integrated. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 173–176, New York, NY, USA, 1991. ACM Press.

[62] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):674–693, 1989.

[63] T. Malzbender. Fourier volume rendering. *ACM Trans. Graph.*, 12(3):233–250, 1993.

[64] A. Mantler and J. Snoeyink. Safe sets for line simplication. *10th Annual Fall Workshop on Computational Geometry*, 2000.

[65] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[66] N. L. Max. Atmospheric illumination and shadows. *ACM SIGGRAPH Computer Graphics*, 20(4):117–123, 1986.

[67] B. H. McCormick, T. A. DeFanti, and M. D. Brown. Visualization in scientific computing. *Computer Graphics*, 21:6, 1987.

[68] R. McMaster and S. Shea. *Generalization in Digital Cartography*. American Association of Geographers, Washington, DC, 1992.

[69] R. B. McMaster. The integration of simplification and smoothing algorithms in line generalization. In *Cartographica*, volume 26, pages 101–121, 1989.

[70] M. Monmonier. *Mapping It Out*. The University Of Chicago Press, 1995.

[71] K. Mueller, N. Shareef, J. Huang, and R. Crawfis. High-quality splatting on rectilinear grids with. efficient culling of occluded voxels. In *IEEE Transactions on Visualization and Computer Graphics*, volume 5, pages 116–134, 1999.

[72] J. C. Müller, R. Weibel, J. P. Lagrange, and F. Salgé. Generalization - state of the art and issues. *GIS and Generalization: Methodology and Practice*, pages 3–17, 1995.

[73] T. Munzner. H3: laying out large directed graphs in 3d hyperbolic space. In *INFOVIS*, pages 2–10, 1997.

[74] S. Muraki. Volume data and wavelet transform. In *IEEE Computer Graphics and Applications*, volume 13, pages 50–56, 1993.

[75] N. Mustafa, E. Koutsofios, S. Krishnan, and S. Venkatasubramanian. Hardware-assisted view-dependent planar map simplification. In *COMPGEOM: Annual ACM Symposium on Computational Geometry*, 2001.

[76] P. Ning and L. Hesselink. Fast volume rendering of compressed data. In *VIS '93: Proceedings of the 4th conference on Visualization '93*, pages 11–18, 1993.

[77] J. Oliensis. Local reproducible smoothing without shrinkage. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):307–312, 1993.

[78] W. Peng and K. Tempfli. An object-oriented design for automated database generalization. In *Proceeding of the 7th. International Symposium on Spatial Data Handling*, pages 199–213, 1996.

[79] J. Perkal. An attempt at objective generalization. *Michigan Inter- University Community of Mathematical Geographers, Discussion Paper 10*, 1966.

[80] B. Peter and R. Weibel. Using vector and raster-based techniques in categorical map generalization. *Third Workshop on Progress in Automated Map Generalization*, 1999.

[81] S. Piccand, R. Noumeir, and E. Paquette. Efficient visualization of volume data sets with region of interest and wavelets. In *SPIE Medical Imaging*, 2005.

[82] R. Rao and S. K. Card. The table lens: Merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *CHI 1994: Conference Proceedings Human Factors in Computing Systems*, pages 318–322, 1994.

[83] A. H. Robinson, J. L. Morrison, P. C. Muehrcke, A. J. Kimerling, and S. C. Guptill. *Elements of Cartography, 6th Edition.* Wiley, New York, 1995.

[84] H. E. Rushmeier and K. E. Torrance. The zonal method for calculating light intensities in the presence of a participating medium. *ACM SIGGRAPH Computer Graphics*, 21(4):293–302, 1987.

[85] G. Sapiro and A. Tannenbaum. Area and length preserving geometric invariant scale-spaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(1):67–72, 1995.

[86] M. Sarkar and M. H. Brown. Graphical fish-eye views of graphs. In *Human Factors in Computing Systems, CHI 92 Conference Proceedings*, pages 83–91, 1992.

[87] M. Sarkar and M. H. Brown. Graphical fisheye views. In *Communications of the ACM*, volume 37, pages 73–83, 1994.

[88] E. Saux. B-spline curve fitting: Application to cartographic generalization of maritime lines. In *8th International Conference on Computer Graphics and Visualization (GraphiCon'98)*, pages 196–203, 1998.

[89] E. Saux. B-spline functions and wavelets for cartographic line generalization. In *Cartography and Geographical Information Science*, volume 30, pages 33–50, 2003.

[90] E. Schwartz, D. Greve, and G. Bonmassar. Space-variant active vision: Definition, overview and examples. *Neural Networks*, 8(7/8):1297–1308, 1995.

[91] E. L. Schwartz. Topographical mapping in primate visual cortex: History, anatomy and computation. In D. H. Kelly, editor, *Visual Science and Engineering*, pages 293–359. Marcel Deckker, New York, 1994.

[92] M. Sester. Generalization based on least squares adjustment. *International Archives of Photogrammetry and Remote Sensing*, XXXIII(B4):931–938, 2000.

[93] P. Shirley and A. Tuckman. A polygonal approximation to direct scalar volume rendering. *Computer graphics*, 24(5):51–58, 1990.

[94] R. Spence and M. Apperley. Data base navigation: An office environment for the professional. *Behaviour and Information Technology*, 1(1):43–54, 1982.

[95] S. Steiniger and S. Meier. Snakes: a technique for line smoothing and displacement in map generalisation. In *7th ICA WORKSHOP on Generalisation and Multiple Representation*, 2004.

[96] M. A. Stricker. Promising directions in active vision. *Int. J. Comput. Vision*, 11(2):109–126, 1993.

[97] G. Taubin. Curve and surface smoothing without shrinkage. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 852, Washington, DC, USA, 1995. IEEE Computer Society.

[98] M. Tistarelli and G. Sandini. On the advantages of polar and log-polar mapping for direct estimation of time-to-impact from optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):401–410, 1992.

[99] H. M. Tong and R. A. Fisher. Progress report on an eye-slaved area-of-interest visual display. *Proc. Image Conference III*, May 1984.

[100] F. Töpfer and W. Pillewizer. The principles of selection: a means of cartographic generalization. *The Cartographic Journal*, 3(1):10–16, 1966.

[101] T. Totsuka and M. Levoy. Frequency domain volume rendering. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 271–278, New York, NY, USA, 1993. ACM Press.

[102] J. A. Tyner. *Introduction to Thematic Cartography.* Englewood Cliffs, New JerseyUSA: Prentice-Hall, 1992.

[103] C. Upson and M. Keeler. V-buffer: visible volume rendering. *SIGGRAPH Comput. Graph.*, 22(4):59–64, 1988.

[104] US Census Bureau. Census 2000 tiger/line data. http://www.esri.com/data/download/census2000_tigerline/, 2000.

[105] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven volume rendering. In *Proceedings of IEEE Visualization'04*, pages 139–145, 2004.

[106] J. Vollmer, R. Mencl, , and H. Müller. Improved laplacian smoothing of noisy surface meshes. In P. Brunet and R. Scopigno, editors, *Computer Graphics Forum (Eurographics '99)*, volume 18(3), pages 131–138. The Eurographics Association and Blackwell Publishers, 1999.

[107] J. M. Ware and C. B. Jones. Conflict reduction in map generalization using iterative improvement. 2(4):383–407, December 1998.

[108] R. Weibel. Generalization of spatial data: Principles and selected algorithms. In *Algorithmic Foundations of Geographic Information Systems*, pages 99–152, 1996.

[109] R. Weibel and G. H. Dutton. Constraint-based automated map generalization. In *Proceedings of the 8 th Interna-tional Symposium on Spatial Data Handling*, pages 214–244, 1998.

[110] C. F. Weiman. Video compression via log polar mapping. In *Proc. SPIE The international society for optical engineering*, volume 1295, pages 266–277, September 1990.

[111] R. Westermann. A multiresolution framework for volume rendering. In *VVS '94: Proceedings of the 1994 symposium on Volume visualization*, pages 51–58, New York, NY, USA, 1994. ACM Press.

[112] L. Westover. Interactive volume rendering. In *VVS '89: Proceedings of the 1989 Chapel Hill workshop on Volume visualization*, pages 9–16, New York, NY, USA, 1989. ACM Press.

[113] L. Westover. Footprint evaluation for volume rendering. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 367–376, New York, NY, USA, 1990. ACM Press.

[114] R. Yagel and Z. H. Shi. Accelerating volume animation by space-leaping. In *VIS '93: Proceedings of the 4th conference on Visualization '93*, pages 62–69, 1993.

[115] B. L. Yeo and B. D. Liu. Volume rendering of DCT-based compressed 3D scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):29–43, 1995.

[116] H. Yu, E. C. Chang, Z. Y. Huang, and Z. J. Zheng. Fast rendering of foveated volumes in wavelet-based representation. In *13th Pacific Conference on Computer Graphics and Applications*, 2005.

[117] J. L. Zhou, A. Döring, and K. D. Tönnies. Distance based enhancement for focal region based volume rendering. In *Proceedings of Bildverarbeitung für die Medizin 2004*, pages 199–203, Berlin, 2004.