ABSTRACT

Title of dissertation: TOWARDS DATA-DRIVEN LARGE-SCALE SCIENTIFIC VISUALIZATION AND EXPLORATION

Cheuk Yiu Ip

Dissertation directed by: Professor Amitabh Varshney
Department of Computer Science

Technological advances have enabled us to acquire extremely large datasets but it remains a challenge to store, process, and extract information from them. This dissertation builds upon recent advances in machine learning, visualization, and user interactions to facilitate exploration of large-scale scientific datasets. First, we use data-driven approaches to computationally identify regions of interest in the datasets. Second, we use visual presentation for effective user comprehension. Third, we provide interactions for human users to integrate domain knowledge and semantic information into this exploration process.

Our research shows how to extract, visualize, and explore informative regions on very large 2D landscape images, 3D volumetric datasets, high-dimensional volumetric mouse brain datasets with thousands of spatially-mapped gene expression profiles, and geospatial trajectories that evolve over time. The contribution of this dissertation include: (1) We introduce a sliding-window saliency model that discovers regions of user interest in very large images; (2) We develop visual segmentation of intensity-gradient histograms

to identify meaningful components from volumetric datasets; (3) We extract boundary surfaces from a wealth of volumetric gene expression mouse brain profiles to personalize the reference brain atlas; (4) We show how to efficiently cluster geospatial trajectories by mapping each sequence of locations to a high-dimensional point with the kernel distance framework.

We aim to discover patterns, relationships, and anomalies that would lead to new scientific, engineering, and medical advances. This work represents one of the first steps toward better visual understanding of large-scale scientific data by combining machine learning and human intelligence.

TOWARDS DATA-DRIVEN LARGE-SCALE SCIENTIFIC VISUALIZATION AND EXPLORATION

by

Cheuk Yiu Ip

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:
Professor Amitabh Varshney, Chair/Advisor
Professor Satyandra K. Gupta, Dean's Representative
Professor David W. Jacobs
Professor Joseph F. JaJa
Dr. David P. Luebke

To the loving memory of my mother, Siu Fong Tse.

Acknowledgments

I would like to acknowledge Dr. Amitabh Varshney for being my advisor. This work will not be possible without his guidance, encouragement, and support. I wish to acknowledge Dr. Joseph JaJa, Dr. David Luebke, Dr. David Jacobs, and Dr. Satyandra K. Gupta for being my dissertation committees and gave me valuable comments on this dissertation work. I also wish to acknowledge Dr. Shankar Krishnan for collaborating with me and gave me valuable advises.

Thanks to my colleagues in my research projects, Sujal Bista, Rob Patro, Derek Juba, and Youngmin Kim. Thanks are also extended to every member of Graphics and Visual Informatics Laboratory during my study. I wish to thank the faculties of the Department of Computer Science for my education in many different aspects of computer science.

I would like to thank my parents for their love and care in my life. I would also like to thank my fellow graduate student friends at the University of Maryland and Johns Hopkins University for supporting me during my study.

This work has been supported in part by the NSF grants: CCF 05-41120, CMMI 08-35572, CNS 09-59979 and the NVIDIA CUDA Center of Excellence. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the NSF or the other supporting organizations.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Big data challenges our capacity to turn data into knowledge for advancing science, engineering, and medicine. This dissertation aims to show that the exploration process of big data can be assisted by integrating machine learning with visualization. We specifically show how users can efficiently identify the regions of interest in datasets by coupling visualization and knowledge discovery. In this dissertation, we analyze and visualize scientific datasets of increasing dimensions, such as spatially large 2D images, 3D volumetric medical and climate datasets, high-dimensional volumetric brain datasets with thousands of spatially-mapped gene expression profiles, and geospatial trajectories that evolve over time.

The advances in computing and acquisition allow us to acquire, process, and store massive amounts of raw data. However, our ability to comprehend the data remains unchanged. Machine learning and visualization can offer help in understanding large datasets in two ways. First, machine learning discovers information from raw data by recognizing patterns, anomalies, and relationships. This artificial intelligence approach reduces the vast amount of data to more manageable knowledge and information. Second, visualization presents the raw data in a visual form for effective user comprehension. Visual representation enhances user cognition and reasoning on the abstract data. User interactions facilitate feedback between machine learning and visualization.

This dissertation shows how machine learning can assist in visual exploration of two, three and multi-dimensional scientific datasets, along with temporal evolution. On very large 2D multi-gigapixel landscape images, we show that our approach of progressive elicitation of salient regions is fast and allows rapid identification of regions of interest. On 3D volumetric datasets, our approach mimics user exploration behavior by ana-

lyzing the intensity-gradient histogram with the normalized-cut multilevel segmentation technique. The resulting segments lead users to discover volumetric regions of interests. We extract gene expression surfaces of different anatomical brain structures from thousands of spatially-mapped gene expression profiles of mouse brains. We also show how to efficiently cluster and detect communities in time-varying geospatial trajectories by using the approximate kernel distance framework.

1.1 The Big Data Challenge

Big data has its origins in scientific disciplines, such as astronomy, to medical discplines, such as genomics, to consumer applications, such as Internet transactions and social networks. The Large Hadron Collider (LHC) generates 15 petabytes of high energy particle experiments data annually [17]. The Sloan Digital Sky Survey (SDSS)[153] has gathered 140 terabytes of optical astronomy images since 2000, its successor, the Large Synoptic Survey Telescope (LSST) [61], is designed to capture 30 terabytes every night with its 3200 megapixel camera. The Expanded Very Large Array (EVLA) [125] radio telescopes are producing 1.2 terabytes of data everyday. Computational methods have played an instrumental role in genomics research. The cost of genomic sequencing has been rapid falling from $100,000, at the completion of the first human genome in 2001, to a few thousand dollars [106] now. This trend is outgrowing the Moore's law and it implies the data can outgrow our computing power in this area. The 1000 Genome Project [148] hosts 130 terabytes of genomics data to provide us a better view on human genetic variation. The Cancer Genome Atlas [107] sequences genomes of tumors and normal pair tissues at a rate of 20 terabytes per month. The Allen Brain Atlas [74] provides a 600 terabytes data archive of how thousands of genes are expressed in different parts of mouse and human brains.

It is imperative for us to understand these massive datasets for enabling the next-generation of scientific discoveries. We need new technologies to analyze, visualize, and

extract useful information from these extremely large datasets. The size of these big datasets challenges the human cognitive capacity, computational capacity, and the visualization capacity. We seek tools for discovering patterns or structures in data and locate regions of interest for visualization. We outline three challenges in big data visualization:

Visual Challenge: The visual challenge arises from the inability of the human visual system to take in all the details that are presented in very large datasets. There are limits on computing both spatial resolution and dimensionality. These arise from a fundamental resolution limitation of the retina as well as the display hardware.

Data Challenge: Processing large amounts of raw data for visualization presents a computational challenge. The increase in dimensionality further exacerbates the problem. We have to address this challenge by better data analysis and the use of parallel computational hardware [117, 118].

Information and Semantics Challenge: Visualization of machine-analyzed data helps users understand the data faster. Identifying informative regions in datasets that match human expectations is an ambitious challenge. We iterate the analysis and visualization with user interactions, such that users can provide valuable semantic inputs to improve the information discovery process.

In this dissertation, we address each of these challenges progressively from 2D images and 3D volumes to datasets with thousands of dimensions and time-varying geospatial locations. The next sections outline the results and the contributions.

1.2 Saliency-Assisted Navigation of Very Large Landscape Images

The field of visualization has addressed navigation of very large datasets, usually meshes and volumes. Significantly less attention has been devoted to the issues surrounding navigation of very large images. In the last few years the explosive growth in the

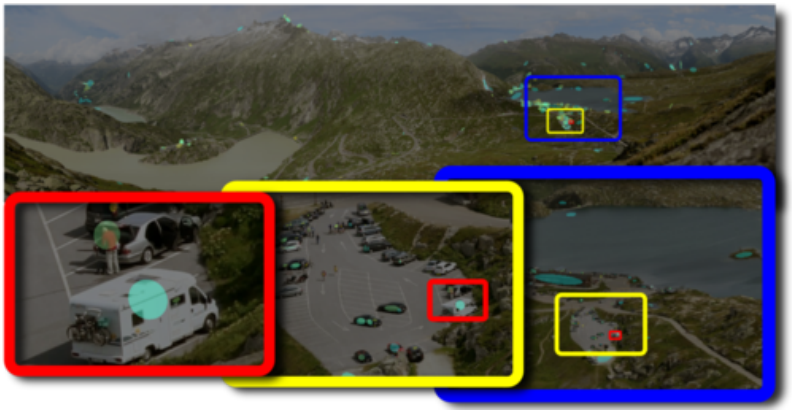


Figure 1.1: Saliency-Assisted Navigation of Very Large Landscape Images: A very large image contains fine details. We progressively zoom into the blue, yellow, and red regions in the panorama. Our sliding-window saliency computation technique simulates this zooming process and detect regions of different scales, such as landscape features, buildings, and human beings for visualization.

resolution of camera sensors and robotic image acquisition techniques has widened the gap between the display and image resolutions to three orders of magnitude or more. This work [56] presents the first steps towards navigation of very large images, particularly landscape images, from an interactive visualization perspective.

The grand challenge in navigation of very large images is identifying regions of potential interest. We outline a three-step approach. In the first step we use multi-scale saliency to narrow down the potential areas of interest. In the second step we outline a method based on statistical signatures to further cull out regions of high conformity. In the final step we allow a user to interactively identify the exceptional regions of high interest that merit further attention. For example, in Figure 1.1, we identify regions of interest, such as landscape features, buildings, and human beings.

We show that our approach of progressive elicitation is fast and allows rapid identification of regions of interest. Unlike previous work in this area, our approach is scalable and computationally reasonable on very large images. We validate the results of our approach by comparing them to user-tagged regions of interest on several very large landscape images from the Internet.

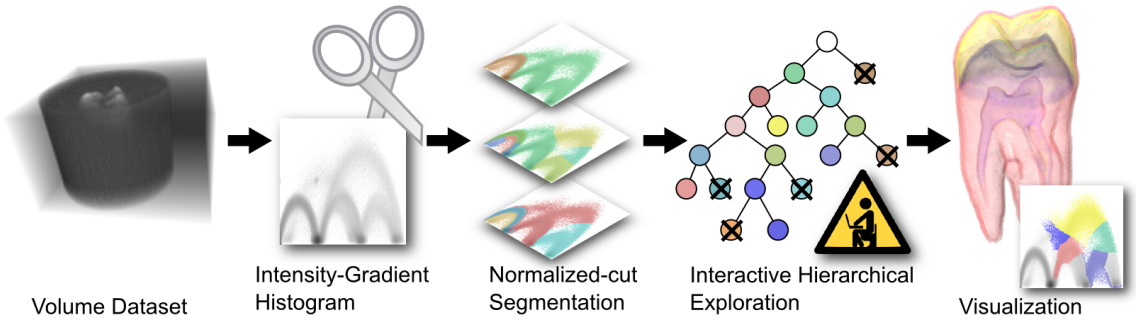


Figure 1.2: Hierarchical Exploration of Volumes using Multilevel Segmentation of the Intensity-Gradient Histograms: We explore a volume dataset with an intensity-gradient histogram segmentation hierarchy. We mimic the user visual search of shapes in the histogram by hierarchically segmenting the histogram image using normalized cuts. Users interactively traverse this hierarchy to discover features in the volume data and compose meaningful visualizations.

1.3 Hierarchical Exploration of Volumes using Multilevel Segmentation of the Intensity-Gradient Histograms

Visual exploration of volumetric datasets to discover the embedded features and spatial structures is a challenging and tedious task. We present a semi-automatic approach to this problem that works by visually segmenting the intensity-gradient 2D histogram of a volumetric dataset into an exploration hierarchy [57]. Our approach mimics user exploration behavior by analyzing the histogram with the normalized-cut multilevel segmentation technique. Unlike previous work in this area, our technique segments the histogram into a reasonable set of intuitive components that are mutually exclusive and collectively exhaustive. We use information-theoretic measures of the volumetric data segments to guide the exploration. This provides a data-driven coarse-to-fine hierarchy for a user to interactively navigate the volume in a meaningful manner.

1.4 Personalizing the Brain Atlas through Gene Expression Correlates

We introduce a data-driven approach to personalize the brain atlas with multiple gene expression profiles [55]. Segmentation of brain volumes is a challenging problem

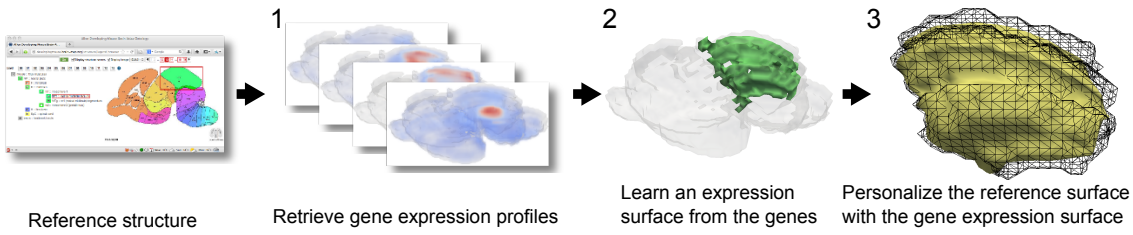


Figure 1.3: Personalizing the Brain Atlas through Gene Expression Correlates: We personalize the brain atlas by learning gene expression surfaces from gene expression profiles. We extract an expression surface from the a set of correlated gene expression profiles in a reference structure. We refine the reference surface to match the extracted gene expression boundary surface.

that often requires fitting a reference atlas with boundaries of the brain structures to a brain volume. Figure 1.3 shows a data-driven approach for extracting these structural boundaries from a wealth of spatially-mapped gene expression scalar fields. We first retrieve genes that are highly correlated with a brain structure of interest. Then, we apply tensor voting to identify a surface with coherent directions that spans the peaks of the gene expression gradients. This 3D surface represents the gene expression boundaries around the brain structure of interest. We visualize the difference between this genetic boundary and the reference atlas. We also compute a deformation that maps the reference atlas to the gene expression surface. This data-driven deformation allows us to personalize the reference brain atlas according to coherent structures in multiple volumetric gene expression profiles. Our experimental results compare a variety of gene expression surfaces to the gene expression profiles and the reference brain structures.

1.5 Fast Trajectory Clustering using the Kernel Distance

We present a novel approach for clustering geospatial trajectories by the approximate kernel distance [54]. Computing a distance between two trajectories is a traditionally challenging problem, due to the quadratic distance computations between points in the two trajectories. The approximate kernel distance method projects point sets, curves, and surfaces to high-dimensional feature vectors for similarity comparison. In this work,

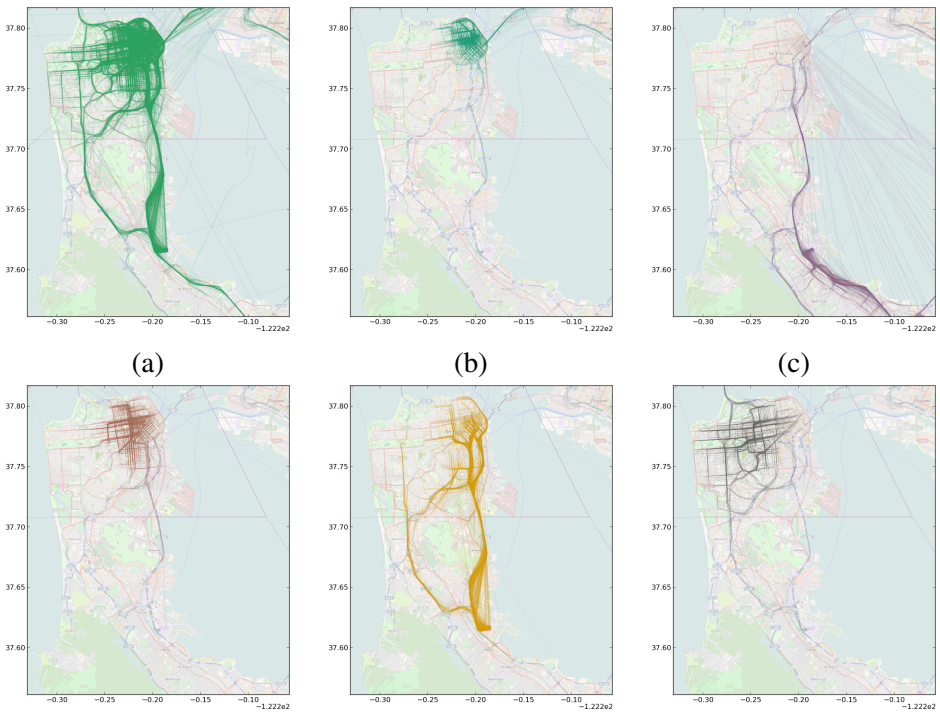


Figure 1.4: Fast Trajectory Clustering using the Kernel Distance: We show how to cluster hundreds of thousands of trajectories with millions of points in minutes. (a) shows 464 K San Francisco taxi trajectories. (b-f) show clusters of popular taxi routes in different parts of San Francisco and surround areas such as the SFO airport in (e).

we show how to map each trajectory to a high dimensional point with this framework in $O(n \lg n)$ for clustering applications. This is very attractive as it allows conventional point-based data mining algorithms to be applied to geospatial trajectories. We present (1) a new technique to cluster geospatial trajectories with an approximate kernel distance, (2) we show how to combine this approach with k-means clustering and operate in a streaming fashion which we can update the feature vectors dynamically as the new points arrive, (3) we also show how to combine the approximate kernel distance framework with community detection methods to automatically discover unevenly distributed clusters of various shapes, and (4) our performance studies show efficient scaling up of this method to process very large datasets with hundreds of thousands of trajectories and tens of millions points within a few minutes. Figure 1.4 shows five clusters of $464K$ San Francisco taxi trajectories.

1.6 Contributions

We outline the contributions of this dissertation for addressing the visual, data, and semantic challenges: visual knowledge discovery, data scalability, as well as, visualization and interactions.

Visual Knowledge Discovery: We use visual approaches to identify regions of interest from visual datasets and provide guided explorations. We simulate the zooming process on very large images by using a sliding window to compute saliency and then visualize the details of a multi-gigapixel image on megapixel displays. On 3D volumetric datasets, we mimic user behaviors by visually segmenting the intensity-gradient histogram of volume datasets. Our work on spatially-mapped gene expression profiles extracts expression surfaces from thousands of gene expression profiles. Clustering similar trajectories geometrically discovers visually similar patterns in time-varying geospatial trajectories.

Data Scalability: We show our visual information discovery can be implemented in a scalable fashion. For spatially large 2D images, we implement sliding-window image saliency computation in an out-of-core fashion on GPUs. We observe that meaningful volumetric segments can be found in the intensity-gradient histogram of a 3D dataset. This 2D histogram is independent of the volume size, thus allowing us to process a large amount of data. We use parallelizable and GPU-friendly techniques, such as tensor voting and isosurface algorithms, to extract expression surfaces from spatially-mapped volumetric gene expression profiles. We enable efficient clustering of time-varying geospatial trajectories by mapping a sequence of points onto a high-dimensional point through random projections. This mapping allows us to apply some highly scalable point-based clustering and community detection algorithms to hundreds of thousands of trajectories within a few minutes.

Interaction and Visualization: We provide efficient visualization and exploration mechanisms for users to associate their knowledge with the data. We visualize small informative regions on very large images and allow users to interactively refine the results. Users hierarchically explore machine segmented components in a volume dataset. We use a surface representation to visualize many coherently expressed genes within a user-specified anatomical brain structure. Clusters and communities of geospatial trajectories allow users to visually identify patterns and spot anomalies.

The remainder of this dissertation is organized as follows: In Chapter 2 we review related work on visual data analysis. In Chapter 3, we show how the computational saliency model can facilitate the navigation in very large landscape images. In Chapter 4, we describe how to explore volume datasets with a multilevel segmentation of intensity-gradient histogram. In Chapter 5, we show how to extract and visualize gene expression surfaces from volumetric brain images with thousands of spatially-mapped gene expression profiles. In Chapter 6, we cluster time-varying geospatial trajectories with the approximate kernel distance framework. Chapter 7 concludes this dissertation and discusses about the directions for future work.

Chapter 2

Related Work

In this chapter, we review the current tools on visual data analysis. We focus on the tools and analysis that are related to our results on large images, volumes, brain images, and geospatial trajectories.

2.1 Overcoming Display Limitations

Over the years a number of techniques have evolved to address limitations of the display medium with respect to the visual data. If the input data has more bits of color information than those that can be displayed, we use tone mapping to approximate the appearance of high-dynamic range images [28]. This involves mapping the color space from a high-dimensional input to a low-dimensional output, while preserving most of the salient content. Similarly, video summarization techniques [26] typically involve extraction of the salient key-frames with some context that results in warping of the temporal space. Recent research has also looked at how to carry out image warping so that images that are larger than the display can be adaptively resized to preserve their most salient content [4, 133]. Most recently, Laffont *et al.* [92] present content-aware zooming on images up to 16 megapixels. While such image re-targeting techniques could be effective for images that are an order of magnitude larger than the displays, they do not scale well to three orders of magnitude or more that we desire. In In Chapter 3 we present distortion-free navigation of very large images, assisted by identification of their most salient content, to allow viewing of very large images on displays of modest sizes. Unlike the previously discussed methods, visualization by navigation maintains the metric fidelity of the image. The pan and zoom interactions allow the users to follow the local image context naturally.

2.2 Scene Analysis and Image Saliency

Image saliency has been used in modeling visual attention. Top-down and bottom-up models for building a saliency map have been introduced by Tsotsos *et al.* [159], Itti *et al.* [60] and several others. Jacobs [62] discusses which viewpoint invariant properties are salient. Suh *et al.* [149] use image saliency to crop thumbnails from images. Oliva *et al.* [116] apply the top-down model to object detection.

Recent approaches include work by Hou and Zhang [50] that computes image saliency by the difference of the image's original and smoothed log-Fourier spectrum. Bruce *et al.* [13] learn a set of sparse code from example images to evaluate saliency of new images. Wang *et al.* [165] use random graph walk on image pixels to compute image saliency. Goferman *et al.* [43] consider visual organization and high level features such as human faces in saliency computation.

In spite of the impressive advances in the recognition of specific objects, such as buildings, cars, and humans, general scene understanding remains a hard problem [47]. State-of-the-art systems [20, 100, 136] train on web-scale databases of small images and then extract regions of interests for test scenes in a supervised manner. More recently, Kim and Torralba [80] use alternating optimization on sets of unlabeled images to extract one to three regions of interest per image. Our system needs a more general approach since we expect a variety of regions and objects of interest in a very large image.

Although there has been extensive previous work in identifying salient regions using several methods, such techniques typically extract a very small number of regions from a relatively small image. For instance, Goferman *et al.*'s [43] program requires 74 seconds to process a 250×142 image, and Bruce *et al.*'s [13] program requires 30 minutes to process a 3000×1500 image. These timings are on a current state-of-the-art workstation described in Section 3.6. Assuming their running time scales linearly and memory swapping does not become an issue, these approaches will take hundreds of hours to process gigapixel-sized images. The purpose of this is not to downplay the achievements of

these approaches, which are actually quite impressive in what they are targeting, rather to highlight the difference in their approaches from ours.

We believe user interaction in the visualization process can greatly assist in rapid culling of false positives and can greatly enhance the overall computational efficiency of the resulting algorithms. Our approach adopts a three-step process of progressive culling of potential regions of interest. We believe this provides a better balance of accuracy and computational efficiency with a user in the middle than a purely computational approach.

2.3 Visual Data Analysis

There is a rich history of data analysis for visual summarization and identification of information-rich subspaces for effective visual presentation. We next present a few example of how salience is defined and used for visual datasets to enhance their depiction. Notable advances on 3D datasets include defining saliency for polygonal meshes [94], comparing it to human eye movements [85], and illuminating meshes based on saliency [93]. Howlett *et al.* [51] use eye-tracking data to identify salient features on meshes and carry out user studies to validate their findings. Kim *et al.* present and validate saliency-based enhancement operators to guide visual attention in volume visualization [83] and geometric meshes [84].

Machiraju *et al.* [104] present a system to detect contextually significant multiscale features in very large datasets directly in the wavelet domain and visualize them progressively. Bordoloi and Shen [12] select informative views of volumetric data based on saliency defined using entropy measures. Viola *et al.* [163] determine the most expressive view for a selected region of interest in a volume using mutual information. Bruckner and Möller [14] use isosurface similarity maps based on mutual information to automatically select the most salient isosurfaces. Saliency-based summarization of time-varying datasets has been carried out for videos [26] and molecular dynamics simulations [123]. Wiebel *et al.* [168] identify salience with the vortices that originate from

walls in three-dimensional time-dependent vector fields and track their evolution using generalized streak lines. More recently, Information contents of datasets have been extensively used to compact representations and visualizations. Jänicke [66, 65] and Chen [64] use evaluate information content to better visualize flows. Kim and JaJa [81] construct isosurfaces from entropy-based octrees.

Unlike much of the previous work on volumetric or time-varying data, the focus of our work is on visualization of very large images.

2.4 Transfer Function Design

Transfer functions directly influence the visualization by assigning optical properties such as color and opacity to voxels. Previous work has devoted a lot of effort on transfer function generation. Pfister *et al.* [126] present a survey of different approaches to transfer function design. Traditionally, a transfer function maps voxels to colors and opacity according to a 1D function of the scalar values. Subsequent work has considered multiple attributes and has involved the design of transfer functions using multiple dimensions. Fujishiro *et al.* [38] and Zhou *et al.* [177] analyze the topology of the scalar field to generate transfer functions. Tzeng *et al.* [160] paint on the volume data to design high-dimensional transfer functions. Sereda *et al.* [141] compute LH histograms to detect material boundaries and transfer functions. Salama *et al.* [137] include abstract semantic attributes to assist in domain-specific visualization design. Correa and Ma [23, 24] have recently incorporate size and visibility into transfer functions. Ruiz *et al.* [135] generate automatic transfer functions based on information divergences.

A popular transfer function attribute is the derivatives of the scalar field, often the gradient of the intensity. Kindlmann and Durkin [86] use the derivatives to show better separation of materials and boundaries. Kniss *et al.* [87] use fixed shape widgets to interactively design 2D transfer functions. Roettger *et al.* [131] create transfer functions by clustering the 2D volume histogram according to the spatial connectivity of the volume

dataset. Users interact with a 1D histogram to manipulate the 2D non-parametric segments in Maciejewski *et al.*'s system [105]. Instead of the volume histograms, Selver and Güzeliç [140] initialize the transfer function by fitting radial basis functions to the histograms of the image slices in a volume dataset. Most recently, Wang *et al.* [167] learn a Gaussian Mixture Model from the volumetric scalar field and use the resulting ellipses to compose transfer functions.

In Chapter 4, we show how to automatically identify segments of interest from the intensity-gradient histograms. We organize these segments from coarse to fine to facilitate the exploration process. Our segments are non-parametric and tightly cover the feature space. Users can interact with these segments directly on the intensity-gradient histograms.

2.5 Visual Data Segmentation

Separating 2D images and 3D volumes into meaningful regions is a long-standing and challenging problem. Here we present a small subset of recent results from the vast literature on this topic. Freixenet *et al.* [37] survey different approaches to segment 2D images. Mean-shift image segmentation [22] treats the image as probability distribution and finds the segmentation according to local maxima. Graph-based segmentation approaches model the pixels as nodes and the image as a connected graph and then they partition the graph to solve the segmentation problem. Felzenszwalb and Huttenlocher [32] use a boundary predicate and a greedy algorithm to segment images. Sharon *et al.* [142] introduce an algebraic multigrid inspired segmentation algorithm.

3D volume segmentation is a natural extension to the 2D image segmentation problem. Huang and Ma [52] apply the region-growing technique to segment volumes. Sherbondy *et al.* [143] use programmable graphics hardware to accelerate volume segmentation. Tzeng and Ma [161] cluster volume datasets with the ISODATA algorithm. Segmentation of various organs from medical images is an important application. Level-

set methods [7, 178] have been used to segment 3D brain images. Grady *et al.* [44] apply random-walk segmentation to detect organs from medical volumes. 2-point and N-point correlation function features [68, 110] have been used to classify 3D tissue images. Fuller *et al.* [39] use support-vector machines to segment retinal layers. Janoos *et al.* [69, 70] segment, reconstruct, and visualize dendritic spines in 3D from optical microscopy imaging. The advances in visual computing technologies have enabled segmentation and visualization of neurons [49, 164] in brains. Computer segmentation systems are often guided by user interactions [115]. Bartz *et al.* [9] use a seed point to segment the bronchi in the lungs. Prassni *et al.* [128] incorporate user guidance to minimize the uncertainties in the brain segmentation. Kniss *et al.* [88] use supervised manifold distance on brain image segmentation.

3D volume segmentation is a natural extension to the 2D image segmentation problem. Huang and Ma [52] apply the region growing technique to volume segmentation. Sherbondy *et al.* [143] accelerate volume segmentation with programmable graphics hardware. Segmentation of organs from medical data is a prime application of volume segmentation. Grady *et al.* [44] apply random walk segmentation to detect organs from medical volumes. Fuller *et al.* [39] use support-vector machines to segment retinal layers. Many systems incorporate user guidance to perform interactive segmentation. Bartz *et al.* [9] use a seeded point to segment the bronchi in the lungs. Prassni *et al.* [128] incorporate user guidance to minimize the uncertainties in the segmentation. Kniss *et al.* [88] use supervised manifold distance to segment volume data. Torsney-Weir *et al.* [158] estimate visual responses to guide the search for the image segmentation parameters.

In Chapter 4, we use the popular normalized-cut approach [144] to directly segment the intensity-gradient histogram of a volume data and construct the transfer function. The normalized-cut approach to 2D image segmentation has been extensively used for a variety of applications and it has been mapped on a variety of architectures including multi-core and many-core GPUs [53].

2.5.1 Atlas-based Segmentation of Brain Images

Segmentation of brain images [34] is an important topic in medical imaging. Atlas-based techniques attempt to register an annotated brain atlas to an input image. The challenge is in devising methods that deform the atlas to fit a single image. Cabezas *et al.* [16] present a recent survey on atlas-based segmentation of MR brain images. When multiple brain atlases are available, selecting the best atlas to fit a particular image is also of interest [48, 132, 170]. Evans *et al.* [31] construct a blurred atlas from hundreds of MR brain images. Ju *et al.* have proposed a geometric framework [76] for landmark-and-atlas-based segmentation [10, 78] on the *in situ* hybridization gene expression images of mouse brains. In this work, we present a method to extract 3D boundaries of a given brain structure from multiple gene expression profiles.

2.5.2 Extracting 3D Surfaces from Scalar Fields

Lorensen and Cline [101] introduced the classical marching cubes algorithm for constructing contour surfaces from a scalar field at a given iso-value. It has been a challenge to locate good iso-surfaces in volumetric scalar fields. Bajaj *et al.* [8] present the contour spectrum tool for users to select iso-values. Kindlmann and Durkin [86] use scalar field gradients to identify better separation of materials and boundaries. Tang and Medioni [154] show how to use tensor voting to extract surfaces from noisy point clouds and scalar fields. Sereda *et al.* [141] introduce LH histograms to detect material boundaries. Bruckner and Möller [15] compute a similarity map of iso-surfaces to determine a good iso-value.

2.6 Clustering Moving Objects

Trajectory clustering techniques group similarly-shaped trajectories or sub-trajectories. Gaffney *et al.* [40] use a probabilistic regression method to cluster the gross shapes of tra-

jectories. Lee *et al.* [95] focus on clustering sub-trajectories by first partitioning the trajectories into line segments and then group the line segments into clusters. Li *et al.* [98] extend this partitioning-and-grouping approach to accommodate incremental datasets. Ferreira *et al.* [33] show how to cluster trajectories by fitting multiple vector fields.

Clustering moving objects at different time steps is another related area. Zhang and Lin [173] use the k-center clustering algorithm to cluster buckets of moving objects. Jensen *et al.* [71] incrementally cluster moving objects by using spatial and velocity information as cluster features.

Discovering groups of moving objects has also been an active research area. Gudmundsson *et al.* [45] find fixed-radius circular flocks of moving objects with a quadtree. Jeung *et al.* [72] relax the fixed-radius requirement and discover convoys that are together in consecutive timesteps by filtering and simplifying the trajectories. Li *et al.* [97] find swarms that are together in non-consecutive timesteps by using forward-backing pruning to reduce the search space. Giannotti *et al.* [42] discover the frequently visited regions of interest from trajectories. Nanni and Pedreschi [112] cluster trajectories with a focus on time. Pelekis *et al.* [124] cluster uncertain trajectories. Tang *et al.* [156] find travel companions from streaming trajectories. Zheng *et al.* [174] use a density-based approach to detect gathering patterns.

Zheng *et al.* have collected a huge dataset of GPS trajectories [175], including thousands of taxi trajectories for improving driving directions [172]. More recently, they have released a new system for taxi ridesharing [103]. Zheng and Zhou [176] have edited a book on computing with spatial trajectories.

2.7 Approximating Complex Distances

We show how we can use a newly developed computational geometry technique called approximate kernel distance to cluster trajectories. In general, kernel methods project input data points into higher dimensions, for better separation of dissimilar points.

This approach has been widely used in machine learning [130, 139] to find non-linear classifiers and underlying subspace manifolds. Similarly, the kernel distance method uses a lifting map to project data points into a higher dimensional space and computes the distances there. This method has been used for comparing point sets, curves, and surfaces. Vaillant and Glaunes [162] have used this method to compare surfaces of human face meshes. Joshi *et al.* [75] have established the bounds for the lifting map requirement. They have also shown how to select a coreset of points and how to estimate translation and rotation to align two different point sets.

Comparing distributions has received significant attention in the machine learning, data mining and computer vision communities. One of the popular distance functions to compare distributions and weighted point sets in general is the Earth Mover Distance (EMD). EMD has been popularized by the computer vision community for comparing histograms in content-based image and video retrieval [134]. Yet computing the Earth mover's distance requires solving a transportation optimization problem with the $O(n^3)$ Hungarian algorithm. Many approximation techniques have been proposed. Ling and Okada [99] empirically show that EMD can be computed in $O(n^2)$ time with L_1 distance. Shirdhonkar and Jacobs [145] compute an approximation to the EMD in $O(n)$ time by using the sum of the absolute weighted wavelet coefficients of the difference histogram. While these approximations work in L_1 space, Andoni *et al.* [2] show how to embed EMD into well-behaved norm spaces of high dimensions. Applegate *et al.* [3] have shown a wavelet approximation for EMD on generalized manifolds.

Chapter 3

Saliency-assisted Navigation of Very Large Landscape Images

We are seeing a significant growth in the interest and relevance of very large images. One of the reasons behind this trend is the development of systems that can automatically capture and stitch photographs to create images of unprecedented detail ranging from a few gigapixels [21, 90, 179] to even a few terapixels [46]. Recent advances in consumer-grade robotic image acquisition from companies such as Gigapan have further energized social network communities that are interested in building, sharing, and collectively exploring such large images. Some relevant work on processing of very large images includes a streaming multigrid solver for gigapixel scale out-of-core gradient-domain image processing by Kazhdan and Hoppe [79]. More recently, Summa *et al.* [150] present progressive processing of high-resolution images with interactive previews. Kopf *et al.* [90] discuss how to naturally display the stitched image by cylindrical projections. Luan *et al.* [102] adaptively annotate these very large images by text and audio according to the viewing position and scale. While these are very interesting first steps in computational processing and display of very large images, this chapter addresses a different challenge for such large images – their effective visual navigation.

Consider a gigapixel image shown in Figure 3.1 . The successive zooms give an indication of the level of detail in such images. When viewing such images, users typically pan at the coarse level and occasionally zoom in to see the fine details. Panning at the finest level of detail is too tedious and panning at the coarsest level of detail does not have enough information for the user to know where to zoom in. Just to convey the magnitude of the problem, let us consider some numbers. Imagine a user is visualizing a 4 Gigapixel image on a 2 Megapixel monitor. This would suggest that every monitor pixel is representing 2000 image pixels and the observable image on the monitor is a mere

0.05% of the total dataset. Further, if it takes a user just a couple of seconds to scan the monitor, it will take more than an hour to scan through the entire image.

In this chapter, we leverage principles of visualization to ease the task of navigating very large landscape images. In visual exploration of very large images the biggest challenge involves identifying the most salient content and visually presenting this information to the user. Just as the transfer function design in traditional visualization uses opacity to identify what data to show and color to emphasize, we present data-driven techniques to identify and emphasize potential areas of interest in very large images. Our techniques are relevant for visualization of datasets when the data size is several orders of magnitude larger than what the display device can accomodate. We use techniques based on visual knowledge discovery to help in user navigation and adaptive context- and scale-dependent visual overlays to assist in spatial localization of salient detail.

Challenges

The interactive visual exploration of very high-resolution large-scale images presents three challenges:

1. Visual Scalability: The visual scalability challenge arises from the inability of the human visual system to take in all the details that are present in a very large image. This arises from a fundamental limitation of the retina as well as the display hardware which have not kept pace with our ability to acquire ever larger images. Figure 3.2 shows the growth in resolution of the mainstream consumer-grade LCD displays against camera sensors in recent years. The display resolutions correspond to the highest-resolution monitors sold by a mainstream vendor (such as Dell and Apple) and the camera-sensor sizes correspond to the highest-resolution entry-level SLR cameras manufactured by Canon, Nikon, or Sony. It is easy to see how the resolution growth of these off-the-shelf cameras has clearly outpaced the resolution of the display monitors.

2. Information Scalability: The challenge here is to design effective computational

Figure 3.1: A very large image contains fine details. We progressively zoom into the blue, yellow, and red regions in the panorama. There are interesting regions at different scales: The overview panorama shows the landscape. The blue region shows the hotel and parking lot. The yellow region shows the cars. The red region shows a human. Note the red region is less than one pixel at the overview scale.

algorithms to identify nuggets of useful visual information that hide in large-scale images. In very large images, most of the image data is innocuous and unimportant and even considering it wastes precious time and resources. Often relatively small regions in such very large images are accorded a very high information value by human observers. Identification of such informative regions in very large images that largely match human expectations is an ambitious challenge.

3. Data Scalability: The sheer data size of these images poses a computational challenge. Processing such large images along with their auxiliary data structures often necessitate out-of-core methods as well as designing of algorithms that are cache- and memory-efficient. Even routine image-processing operations for very large images require a careful mapping to the many-core and multi-core processor architectures for any reasonable performance.

Contributions We present the first steps towards addressing the above challenges for interactive visual exploration of very large images. We outline our main contributions:

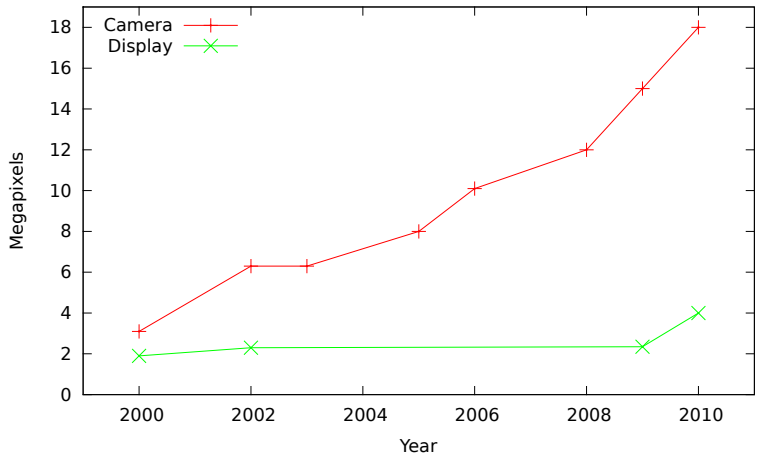


Figure 3.2: The growth of camera sensor resolution vs display resolution.

1. We extend classical computational image saliency to very large images. Users often navigate across three or more orders of magnitude scale differences – from the overview to the finest-scale views while viewing a very large image. Classical algorithms for multi-scale image saliency break down at handling such a large span of visual scales. We discuss the issues involved and present a solution in Section 3.2.

2. As discussed above in the information scalability challenge, it is important to identify the regions of interest in very large images that characterize areas of high information value to users. The question of how to effectively characterize visual information content is still far from settled. There are a number of measures of visual information content and often the definition depends on the task at hand. Our goal is not to provide a definitive characterization of the visual information content of a region of an image, which is a very deep question related to the issues of task semantics and knowledge. Instead we present here a fairly general information discovery algorithm in Section 3.3, that can serve as a framework for further research with other measures of information content.

3. Interactive visual exploration of very large images requires a careful balancing of computational analysis and user preferences. Too much reliance on automatic

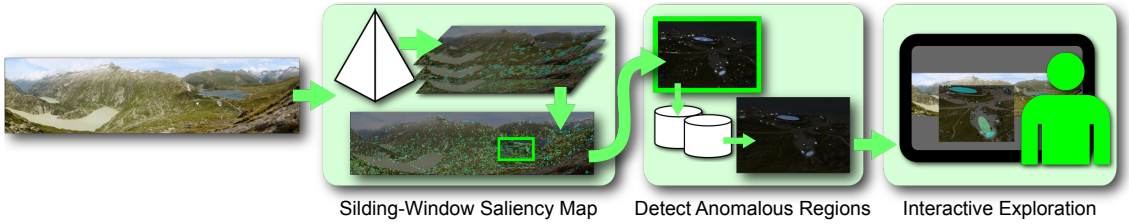


Figure 3.3: Our system discovers regions of interests in very large images and assists user exploration. In the first step we build a saliency map by augmenting the traditional multi-scale image saliency approach with a sliding window over scales. In the second step we carry out information discovery by using color descriptors to identify the most unique regions. In the third step we facilitate rapid elimination of false positives through user interaction during visualization.

intelligence-extraction algorithms is currently not feasible since it is often very difficult to codify semantics of what a user is looking for. At the same time a purely interactive visual exploration without any computational assistance proves to be tedious and overwhelming due to the sheer scope of the data that is being visualized. We present an interactive visual exploration and information discovery system in Section 3.4.

4. Data scalability is an important issue when dealing with very large images and we present advances in this area in Section 3.5.

5. We present and compare our results with those from a social community of gigapixel image enthusiasts in Section 3.6.

3.1 Overview

The goal of this chapter is to carry out computationally-assisted navigation of very large images. We leverage the principles of visual saliency and statistical similarity to outline a three-step approach. Fig. 3.3 shows an overview of our approach.

Our first goal is to identify regions of interesting detail in very large images. The challenges here are in dealing with the dramatic span of visual scales and the sheer amount

of data as well as information. Our approach addresses each of them.

Visual Scalability: Traditional algorithms for multi-scale image saliency work well for small images up to a few megapixels but do not scale up well to gigapixels and beyond. To address this we augment the traditional multi-scale image saliency approach with a sliding window over scales to effectively work with very large images. Our approach only requires Gaussian convolutions on images. It is highly parallelizable and scales linearly with the size of the image. The sliding-window saliency map phase of our approach discovers thousands of locally salient regions from billions of pixels.

Information Scalability: Typical landscape images comprise of a large number of natural elements such as clouds, rocks, grass, and trees. In this chapter we assume that such repeating scene elements are not of interest to the viewers. We characterize all image regions using automatic color-structure descriptors. We then argue that the most interesting regions are the ones that are the most different from their k nearest neighbors (k-NN) in such color-structure feature space. We have empirically observed that this definition works well for landscape images. Other descriptors may be found to be more suitable for other datasets. We use a spatial index to accelerate the k-nearest-neighbor queries. This indexing and querying process grows as $O(n \log n)$, where n is the number of salient regions identified by the sliding-window saliency step. For gigapixel images n is typically of the order of a few tens of thousands. We refer to this step as *anomaly detection*.

Interactive Visual Exploration: We have developed an interactive visualization environment to assist in exploration of very large images. We facilitate users to be aware of details that are a fraction of the screen pixel by ensuring that the overlays for such regions are large enough to be visible at every scale. The users can then explore and inspect all such regions interactively. We also have an automatic mode of the system in which the users are led through a smooth camera fly-through over all the informative regions in the

Figure 3.4: We zoom into a very large image (a) to see the lake and the nearby terrain, (b) to discover the hotel and parking lot (b) and then go still further (c) to see the cars in the parking lot.

image in order of their uniqueness as determined by the Visual and Information Scalability stages of the algorithm. If a user comes across a region of the image that the system claims is important, but the user finds to be unimportant, the user can identify all similar regions (using a slider) and discard them interactively. The spatial index structure built to address the Information Scalability stage of the algorithm allows this operation to occur within a few milliseconds.

Data Scalability: The size of the very large images together with their auxiliary data-structures imposes a significant computational and storage burden. We have used a number of approaches to ameliorate this problem including out-of-core computation, efficient storage of salient regions, and building the a clipmap-based image viewer.

3.2 Sliding-Window Saliency

A very high resolution image is particularly interesting because it exceeds what the human eye can see at a given spot. In a very large image we can zoom from seeing the big picture to scrutinizing the finest details. Figure 3.4 shows three different views of an image at three different scales. We note that the cars seen in Figure 3.4(c) are not discernible in Figure 3.4(a).

3.2.1 Traditional Image Saliency

A saliency map shows which part of an image is likely to attract the most attention of the low-level human visual system. Itti *et al.* [60] have proposed a computational model of visual saliency by using multiscale image processing. Multiscale image processing techniques analyze an image at different scales to simulate the retinal receptive fields. Their image saliency model aggregates the results from three features of an image – intensity, color opponencies, and orientation. We have found that the use of the orientation features decreases the quality of our results as it ends up enhancing naturally occurring structures with strong edges such as cracks in rocks or trees, that end up becoming too salient. In this chapter we only consider the intensity and the two color-opponency attributes for computing image saliency. The intensity (F_I) is the average of primary colors, red, green, and blue. The color opponency attribute contains two sub channels, Red-Green(F_R) and Blue-Yellow(F_B). The details on computation of these attributes can be found in [60].

We next briefly review the traditional algorithm for computing the saliency map $\mathcal{S}$ of an image.

$$
\begin{aligned}
F_i &\leftarrow \text{Image}, \quad i \in \{I,R,B\} \\
G_{i,j} &= \mathcal{G}(j) \otimes F_i, \quad j \in \sigma, 2\sigma, 4\sigma, 8\sigma, 16\sigma \cdots \\
D_{i,j,k} &= |G_{i,j} - G_{i,k}|, \quad k \in \{4j, 8j\} \\
N_I &= \sum_k \sum_j \mathcal{N}(D_{I,j,k}) \\
N_C &= \sum_k \sum_j [\mathcal{N}(D_{R,j,k}) + \mathcal{N}(D_{B,j,k})] \\
\mathcal{S} &= \frac{1}{2}[\mathcal{N}(N_I) + \mathcal{N}(N_c)]
\end{aligned} \tag{3.1}
$$

We first extract the intensity feature, F_I, and color features F_R, F_B from an image.

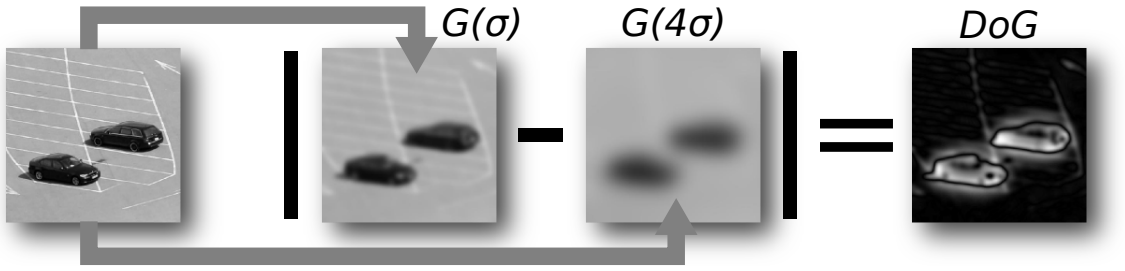


Figure 3.5: Computational saliency mimics the contrast detection mechanism in the human retina. This image shows how Difference of Gaussians (*DoG*) operator detects the high contrast cars instead of the low contrast parking grids.

Then, we convolve the feature images F_i with Gaussian kernels, $\mathcal{G}$, at different scales j. We find contrasting regions by computing the difference of Gaussians (*DoG*) images at each scale, $G_{i,j}$. We compute the *DoG* images at scales $\{\sigma, 4\sigma\}, \{\sigma, 8\sigma\}$. The *DoG* operation mimics the contrast detecting receptive fields of retinal ganglion cells. Figure 3.5 shows the *DoG* operation extracts contrasting cars from the background. $\mathcal{N}$ is a normalization function that promotes the peak salient regions [59]. The saliency map, $\mathcal{S}$, is an aggregation of the normalized *DoG* images. We use $\sigma = 2.0$ in our experiments.

3.2.2 Sliding Window Aggregation

To see the difficulties introduced by the traditional saliency method, consider the saliency map for the parking lot image in Figure 3.4 computed by Itti's *et al.* [60] method. If we aggregate the saliency at all the levels of detail, we obtain the salient region in Figure 3.6(a). We observe that mostly the center of the parking lot has been included, while most of the surrounding cars have been excluded. It is interesting to note that if we analyze the saliency maps at each scale we find that cars are salient at fine scales $\sigma, 2\sigma, 4\sigma$ and the parking lot is salient at scales $16\sigma, 32\sigma, 64\sigma$. This can be seen in Figures 3.6(b)–(d). Therefore even though the saliency maps at individual scales were able to correctly identify the constituent salient elements, the overall aggregation ended up suppressing a number of them. The reason behind this is that if the salient regions at

two different scales overlap, this overlap tends to disproportionately promote the overall salience of that region. Such events are infrequent or otherwise are not of great concern when the image sizes are relatively small. However, this no longer holds true in very large images. As shown in Figure 3.4, an observer would recognize the parking lot and the cars independently of each other at different scales. Therefore we believe that it is inappropriate to aggregate the saliency of the cars and the parking lot together since they are detected by *DoG* filters that are 16 scales of difference apart. The key observation here is that while simulating the multiscale capabilities of the human visual system, we have to be aware that our eyes have a finite resolution. Therefore we should limit the number of scales in multiscale image processing based on the limits of the human visual system.

To address the above, we introduce a sliding-window approach to build saliency maps at multiple scales with limited aggregation. In this approach by limiting the scales of saliency aggregation we produce multiple maps that simulate the zooming operation. This allows views of drastically different scales to be analyzed virtually independently of each other. For example, we can extract cars from the aggregation of normalized maps at scales $\{\sigma, 2\sigma, 4\sigma\}$. Figure 3.6(b) shows the resulting saliency map. It highlights most of the cars without highlighting the parking lot. We separate salient regions at widely different scales by limiting the aggregation procedure.

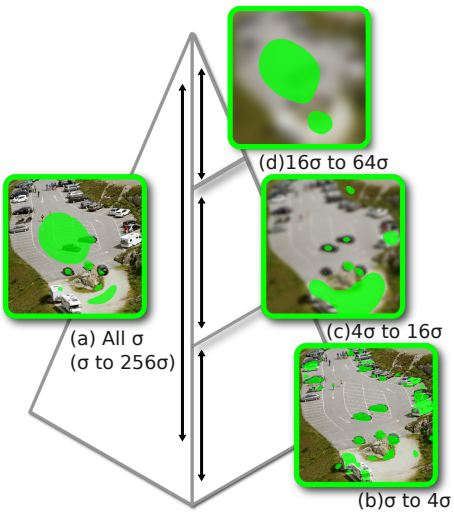


Figure 3.6: This shows a comparison of the thresholded Itti *et al.*'s [60] method in (a) and our sliding window saliency maps in (b)-(d). (a) The center cars and the parking lot are salient while the surrounding cars are suppressed (σ to 256σ). (b) All cars are salient (σ to 4σ). (c) The lower part of the parking lot becomes salient with a few cars (4σ to 16σ). (d) Only the parking lot is salient (16σ to 64σ).

In general, we limit the aggregation of

normalized maps from scale j to scale $j+\delta$. We modify the aggregation procedure in equation (3.1):

$$
\begin{aligned}
N_{I,j} &= \sum_{k} \sum_{j}^{j+\delta} \mathcal{N}(D_{I,j,k}) \\
N_{C,j} &= \sum_{k} \sum_{j}^{j+\delta} [\mathcal{N}(D_{R,j,k}) + \mathcal{N}(D_{B,j,k})] \\
\mathcal{S}_j &= \frac{1}{2}[\mathcal{N}(N_{I,j}) + \mathcal{N}(N_{C,j})]
\end{aligned} \tag{3.2}
$$

This sliding window approach ensures overlapping regions from drastically different scales do not interfere with one another. We seek a scale difference δ that truly reflects the human visual system. We use $\delta = 4j$ in this chapter. We next use arguments from the human visual system theory to suggest why this may be appropriate.

Human Visual System Considerations We would like to use the scales in multiscale image processing based on the sensitivity difference between foveal and peripheral vision. Perceptual studies have shown that the fovea is the most sensitive region of the retina and the sensitivity drops as the view angle increases. Let us assume that we are viewing an image on a 30-inch monitor at 1 meter. In this case, the view angle subtended from the edge of the monitor to the center of the screen is about 20°. At 20°, our retina retains approximately $1/5^{th}$ of the foveal resolution. Therefore, we have decided to compute the saliency maps with images within the 4 scales ($\sigma - 4\sigma$).

Figure 3.7 shows the saliency map resulting from our sliding-window-scale approach for an example image. The computational saliency model detects 18 thousand salient regions. The computational saliency model pre-processes the data efficiently and reduces our quest for interesting and meaningful detail from billions of pixels to thousands of regions. Yet this is still too many regions for a user to manually inspect. We next discuss a more discriminating anomaly detection procedure to refine this initial pool of candidate regions.

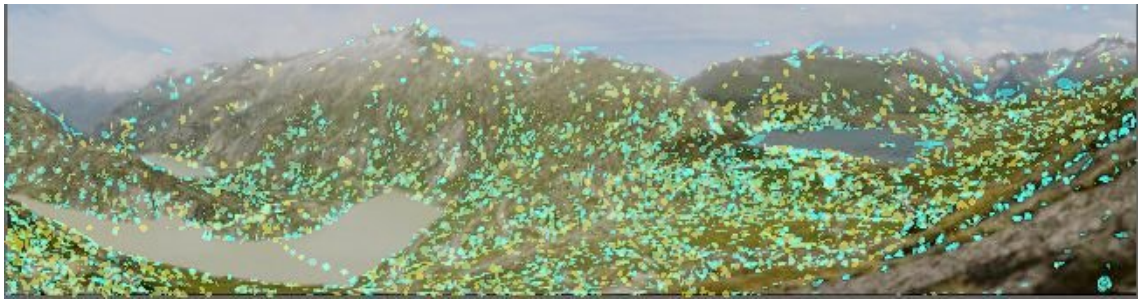


Figure 3.7: The saliency map of our example image. The sliding-window saliency map detects 18k regions. (The salient regions are enlarged for visibility)

3.3 Information Discovery

It is difficult to quantify the visual information content of a region. Some very interesting advances have been made in this field recently. Jänicke *et al.* [66, 65, 67] have extended the idea of local statistical complexity to measure the local information content of a region. This provides an application-independent, purely mathematical measurement of information. More recently, a very interesting and expansive treatment of how saliency and information theory can be adapted and adopted for visualization has been carried out by Chen and Jänicke [18, 64]. In this chapter we wish to slightly side-step the deeply intriguing topic of how to quantify visual information content of a region in the general case, and instead talk about what we have found to work well for large-scale landscape images. We hope that further advances in the field of visual information quantification will seamlessly replace the method that we outline next to work with a wide variety of very large image databases beyond just landscape images.

In this chapter we have decided to adopt the approach that the most important regions of an image are those that are the most different from every other region. In other words, we are interested in identifying the outliers, or the anomalies, in a very large image. As seen in Figure 3.7, the sliding-window saliency aggregation step identifies a large number of salient regions. They range from patches of grass on the ground, to cracks between the rocks in the mountains.

3.3.1 Image Region Descriptors

To be able to quantify differences amongst different image regions we need to first identify what we mean by an image *region* and then we need to settle on the space in which such differences will be measured.

To identify an image region, we fit oriented ellipses to the salient regions that were detected in the previous section. We then fit bounding boxes to the ellipses, pad them by a few extra pixels (we have used 20 pixels for all the examples in this chapter) to ensure that the padded bounding boxes fully enclose the salient regions. These bounding boxes then represent the image regions of interest.

To achieve rotational invariance, we use histograms of the color-space of the pixels belonging to the region of interest. We have tested a number of color spaces – RGB, HSV, CIELab and found that neither of them were very discriminative. We also experimented with shape and orientation descriptors and found that they were excessively discriminative. This search led us towards a descriptor that would represent both color as well as statistical structural information of an image region and would have a discriminating ability that would lie between the two extremes (color-based and edge-based descriptors).

The MPEG-7 color-structure image descriptor represents both color and structural information. The MPEG-7 is an ISO standard for describing multimedia content data and facilitates information retrieval. It consists of generic descriptors that cover many basic visual features, such as color, texture, and shape. The MPEG-7 color-structure descriptor embeds color structure information into the descriptor by counting color frequencies in a moving window of 8×8 pixels. Color values are represented in the double-coned HMMD color space, which is quantized non-uniformly into 64 bins. The range of histogram is normalized to $0-255$. The resulting descriptor is a 64-dimensional vector. We follow the recommendation of MPEG-7 standard and compare them using the Euclidean L_2 norm distance.

$$D(p,q) = \|p-q\|_2$$

p and q are the descriptors of two image patches and $D(p,q)$ is the distance in between the descriptors. We compute the MPEG-7 color-structure descriptor for each image patch using the software provided by the `BilVideo-7` [5] video indexing and retrieval system.

3.3.2 k-Nearest-Neighbors Anomaly Detection

We compute the uniqueness of each region by considering its k-nearest-neighbors (k-NN). For each image region, we search for its k-nearest-neighbor regions:

$$U(p) = \sum_{i=1}^{k} \frac{D(p,q_i)}{k} \quad \text{where} \quad p,q_i \in P, p \neq q_i$$
$$D(p,q_1) \leq D(p,q_2) \leq \ldots \leq D(p,q_k) \leq D(p,q_{k+1}) \ldots \tag{3.3}$$

P is a set of image patch descriptors. The uniqueness of image patch p, $U(p)$, is its average distance to its k-nearest neighbors, $q_1 \ldots q_k$. Repetitive regions with many close neighbors have a low average distance. Regions such as humans, signs, or vehicles should be distinct from the other regions and have a high average distance. We identify the unique regions of interest by their high average distances. We select the top 3% of salient regions as the regions of interest in our experiments.

Approximate nearest-neighbor data structures accelerate the k-nearest-neighbor search [138]. The biggest overhead in the k-nearest-neighbors anomaly detection is the need to retrieve k-nearest-neighbors for each region. Linear search of the k-nearest-neighbor queries is computationally expensive. Research in computational geometry provides many spatial data structures to facilitate this nearest neighbor querying. The approximated nearest-neighbors index significantly accelerates this search process. The sum of distances to the top k-approximated nearest-neighbors provides a reliable uniqueness estimate of each region. Our implementation uses a randomized KD-Tree index in the `flann` library [111] through the `OpenCV` library.

Figure 3.8(b) shows an image with detected regions overlaid after anomaly de-

tection phase. This process reduces the 18 thousand salient regions to just about 500 anomalous regions.

3.4 Interactive Visualization

We guide users to explore the image through the detected regions and interactive visualization. We visualize the detected regions, provide automatic fly-through, and allow interactive user refinement. We highlight three features to assist large image exploration.

- Adaptive scaling ensures the regions of interest are visible.
- Automatic exploration guides the user to discover the unique regions of the image.
- User interaction refines the computed detections.

3.4.1 Visualizing the Detected Regions

We want the users to see the detected regions from the macro view of the image and also allow them to zoom-in to inspect. We believe maintaining the zooming procedure gives the users a much more natural context. Small regions are invisible at the macro view, therefore the corresponding overlay regions are also too small to be seen. We need to visualize these regions more effectively.

We adaptively scale the overlay tags on the detected regions to ensure the most unique regions are visible. We compute the scale, λ, as follows:

$$\lambda(r) = \begin{cases} \frac{s_m}{s_r}\left(1 - \frac{\text{rank}(r)}{\#\text{regions}}\right) & \text{if } s_r < s_m \\ 1 & \text{otherwise} \end{cases}$$

$\lambda(r)$ aims to enlarge the overlay tag for the region r according to the viewing scale and the uniqueness of region r. s_r is region r's size in pixels on the screen. s_m is a user-defined target screen size for the overlay tag. $Rank(r)$ is the relative order of the

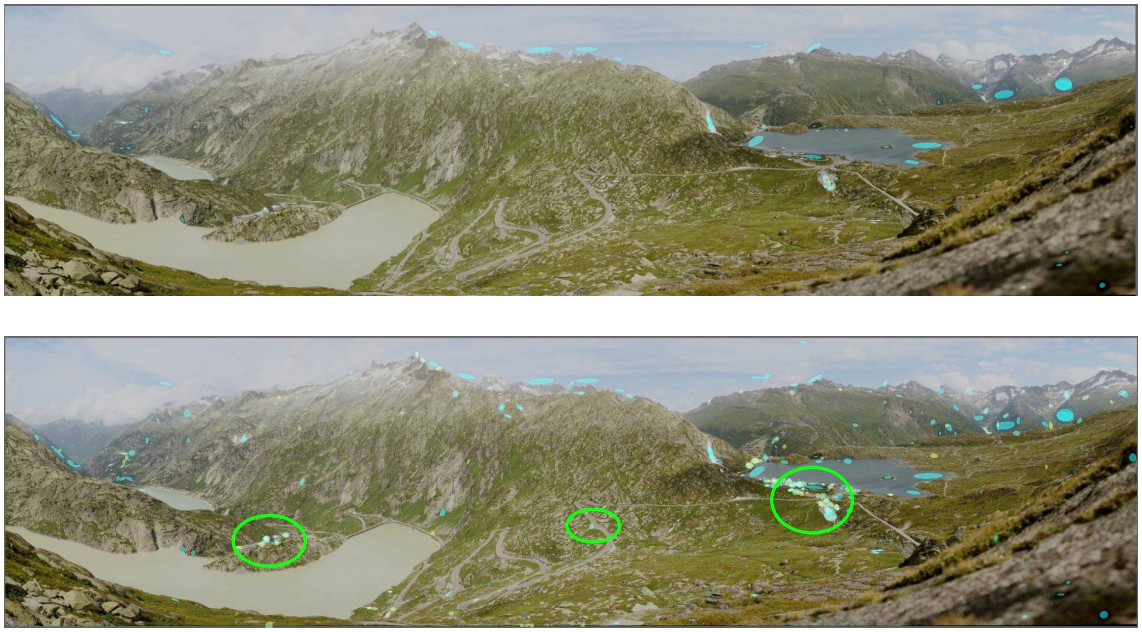


Figure 3.8: We visualize the detected regions with adaptive scaling. (a) Small detected regions are too small to be seen in the macro view. (b) Adaptive scaling ensures the unique regions are enlarged and visible.

uniqueness of region r as determined by equation 3.3 and # *regions* is the number of regions identified at the end of the anomaly detection phase. Thus, the greater the uniqueness of a region r, the greater the $\lambda(r)$ would be. As we zoom into an image, s_r increases and $\lambda(r)$ gradually decreases, and the overlay tags will be shown in their actual size once $s_r \geq s_m$. We color the overlay tags from cyan to magenta (the most unique) according to their rank of uniqueness (equation 3.3).

Figure 3.8 shows the example image and the overlay tags on the detected regions. We see very few detected regions in Figure 3.8(a) because they are too small to be seen. We scale the detected regions in Figure 3.8(b) by $\lambda(r)$. Figure 3.8 (b) shows two groups of regions on the left and on the right. The center region corresponds to a single salient car on the road.

3.4.2 Automatic Exploration

Our system guides the users to fly through the detected regions. This helps the users to start exploring the image when they know little about it. It smoothly pans and zooms into the detected regions according to their uniqueness. We achieve this by sorting the regions in descending order of their uniqueness (equation 3.3). This forms a natural exploration sequence that starts from the most unique regions. If users come across misidentified regions during the fly-through, they can suppress the regions by interactive refinement.

3.4.3 Interactive Refinement

Automated systems recognize a lot of regions of interest but they may also make mistakes. Our system allows the users to refine the results by selecting misidentified regions and deleting them. This mechanism allows the users to refine the automatic result but it can be tedious when users need to delete multiple regions. We provide interactions in our system to batch delete misidentified similar regions.

The user may delete a batch of misidentified regions that are similar in a single interaction. Figure 3.9 illustrates this mechanism. This select-slide-delete mechanism can remove many misidentified similar regions at once. In the first step the user identifies a set of regions that in the opinion of the user have been misidentified by the system. Amongst these regions, the user selects one representative region in the second step. This is highlighted by the system. In the third step the user adjusts a slider control to change the similarity distance from the one misidentified representative region to others that are like it. The system interactively highlights other regions that it detects to be similar to the misidentified region. Once the highlighted regions match the user's intent, they can be deleted all at once.

The spatial index and color-structure descriptors of regions provide the interactive search capability. The anomaly detection spatial index is used in this step to provide fast

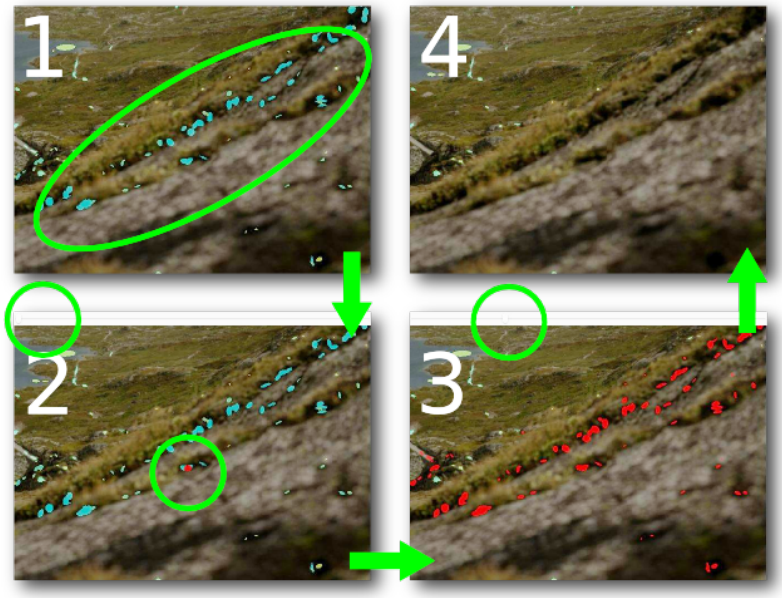


Figure 3.9: Users refine the results by deleting misidentified regions in batch. 1. Locate similar misidentified regions. 2. Select one representative. 3. Adjust the slider to cover the desired regions. 4. Delete the selection.

similarity queries for regions. The system expands the region selection by querying the spatial index for regions that are similar to the user selection. The slider thresholds the number of regions retrieved. The spatial index performs a fast nearest-neighbor query to retrieve similar regions. The system includes these regions in the selection and performs these operations at an interactive speed. Figure 3.10 shows our results after a few user interactions. The number of detected regions reduces from 500 to about 300 with just three such interactions.

3.5 Data Scalability

Processing many gigabytes of image data requires out-of-core algorithms and techniques. We pay special attention to memory constraints when we implement our saliency computation, storage of results, and our interactive viewer.

Figure 3.10: A few user-refinement interactions remove most of the misidentified regions. The number of regions reduces from 500 to about 300 with three select-slide-delete interactions.

3.5.1 Out-of-core GPU Saliency Computation

We use GPUs to accelerate image saliency computation. The required operations for Difference of Gaussians (*DoG*): image filtering, addition, subtraction, and resizing are highly parallelizable and suitable for GPU implementation.

We compute a Gaussian pyramid to approximate the Gaussian blurred images at different scales. We repeatedly downsize the image by a scale of two and convolve it with a fixed Gaussian kernel of scale σ. We store the $x\sigma$ scale Gaussian image at level $\log_2(x)$ of the pyramid. The *DoG* images can be computed by simply finding the difference of Gaussians images between two levels. The non-linear normalization function $\mathcal{N}$ iteratively applies the *DoG* operation [59] to promote the peak regions. We compute each of these normalized maps once and use them to compose sliding-window saliency maps .

A significant problem in processing very large images (which currently are a few gigapixels) is that the entire image will not fit in the GPU or main memory. To address this, we divide each image into small tiles (256×256). We load the image tiles into the GPU independently for addition, subtraction, and resizing operations.

Gaussian filtering requires information on image boundaries. Filtering each tile without overlap results in loss of information at the tile boundaries. To address this we load these tiles into the GPU with overlaps for filtering. For every sub-image that is to be filtered we load two extra rows of tiles (top and bottom) and two extra columns

of tiles (left and right) that surround the sub-image. We fill the GPU memory with the largest possible sub-image (with additional surrounding overlapping tiles) to ensure efficient processing and minimize re-filtering of overlapping tiles. Depending on the loading order, either one row or one column will be used for filtering the next consecutive set of tiles. The ability to independently filter these tiles allows parallelization.

In our implementation, we use the NVIDIA performance primitives for GPU Gaussian filtering, resizing, addition, and subtraction.

3.5.2 Salient Regions Storage

We store ellipses to approximate salient regions to ease storage. The sliding-windows saliency maps incur a storage burden. Similar to storing mipmaps, it takes $1\frac{1}{3}$ times the image size to store the saliency maps at all levels. Although it is a constant factor increase, doubling the storage of the already very large images poses a challenge.

To address this, we first threshold the sliding-window saliency maps and locate continuous regions, also sometimes referred to as *blobs*. The open-source library `cvblob` provides blob detection in our implementation. We carry out local Principal Component Analysis (PCA) to fit ellipses to these regions. This allows us to reduce the storage of each region from hundreds of pixels to a few parameters (center, orientation, and principal axes intercepts of each ellipse). The threshold of our experiments is 0.25.

3.5.3 Tiled Image Viewer

Very large images presented in this chapter do not fit in the GPU or main memory for display. Each gigapixel in RGB format takes three gigabytes of memory in an uncompressed format for rendering. Images with even a few gigapixels exceed the GPU or main memory of a workstation. We have implemented an out-of-core tiled-image viewer. Our viewer fetches only what can be seen at an appropriate scale from the disk. This is similar in spirit to the concept of a clipmap [157]. To carry this out we build a mipmap pyramid

of the image. We divide the images on each level into tiles of 256×256. We load the required image tiles according to the viewing parameters. We pre-fetch two extra rows and columns of image tiles surrounding the viewing region to provide smooth panning. Loading images from the immediately nearby scales prepares for the zooming operation. We store these images in a texture array. This array is independent of the display arrangement of the textures. We map each texture to an array location by a hash function. The loaded texture can be reused on different views without any memory movement. The memory usage of the viewer is independent of the size of the image. It is related to the size of the viewing window on the screen only. Our viewer needs 350 megabytes for viewing a five gigapixel image with a few hundred overlay regions.

3.6 Results

We evaluate our approach on four multi-gigapixel images. We report the regions identified by our system and compare them against web community tags. We also report timings of our experiments. We perform our experiments on the Linux platform with one Intel E5420 CPU, 4GB RAM, and one NVIDIA GeForce GTX 295 GPU (895MB).

3.6.1 Datasets

Digital image stitching and consumer grade robotics have made panoramic photography very popular. Compact digital cameras can stitch multiple consecutive pictures into a panorama. Robotic devices such as the Gigapan EPIC can take hundreds of pictures automatically for image stitching. These products allow consumers to create images of several gigapixels. Community panorama websites such as Gigapan and HDView have gained much popularity on the Internet. Users around the world upload their panoramic pictures to these websites. The web community of panoramic photography enthusiasts then explore, tag, and comment on interesting regions in these images. We downloaded

four gigapixel images from the Gigapan website as shown in Figure 3.11, 3.12, and discussed below.

Grimsel Pass : The Grimsel Pass is a high mountain pass in Switzerland. It connects the valley of Rhone River in the canton of Valais and the Haslital in the canton of Bern. The picture shows an overview of the mountain area, the lake, and the road network. There are distinctive areas with building constructions, hotels, and numerous cars on the road.

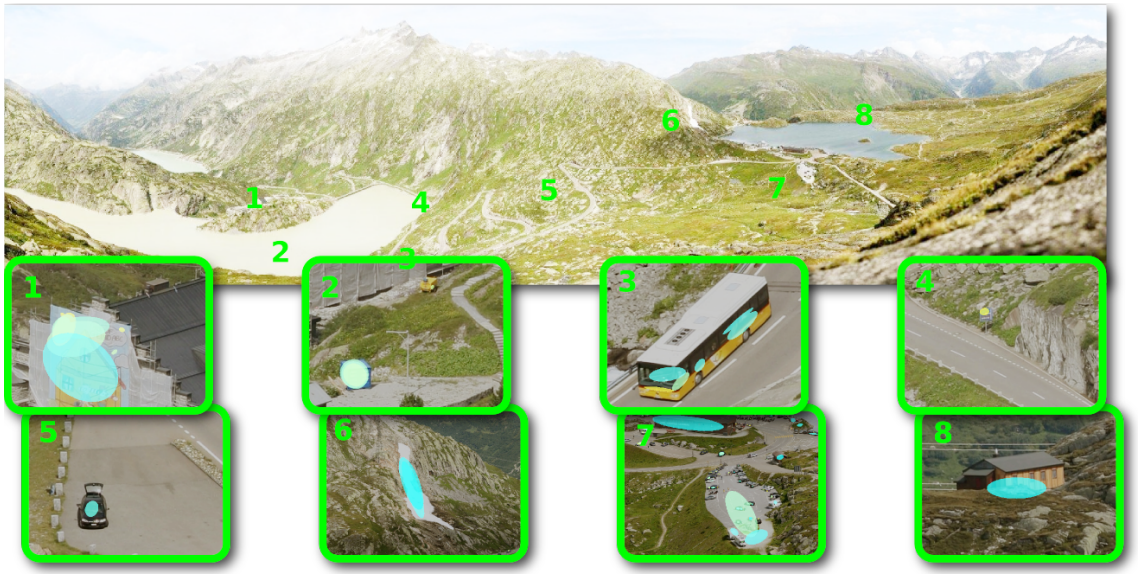
Royal Gorge Bridge : The Royal Gorge Bridge is the highest suspension bridge in the world. This tourist attraction is located in a theme park near Canon City, Colorado, shown in the top right of the picture. The Royal Gorge Route Railroad is a heritage railroad offering scenic and historical train-rides, shown at the bottom left of the picture. The picture shows the valley of Arkansas River with the Royal Gorge Bridge and a small town on the right.

Cacti : This picture shows a cactus field in Arizona. A few hikers are hidden among thousands of cacti.

Main Mt. Whitney Trail : This is a trail in the Sequoia National Park, California. This image of mountains and lakes includes many hikers. They all took the challenge to hike the highest peak (14,497') in the lower 48 states.

3.6.2 Evaluation

We show a sample of the detected regions in Figure 3.11 and 3.11. We tag the locations with numbers and show the detected region in the corresponding thumbnails. We overlay the ellipses onto detected regions. Our system identifies a variety of regions at multiple scales. It locates humans, vehicles, buildings and even special features of the

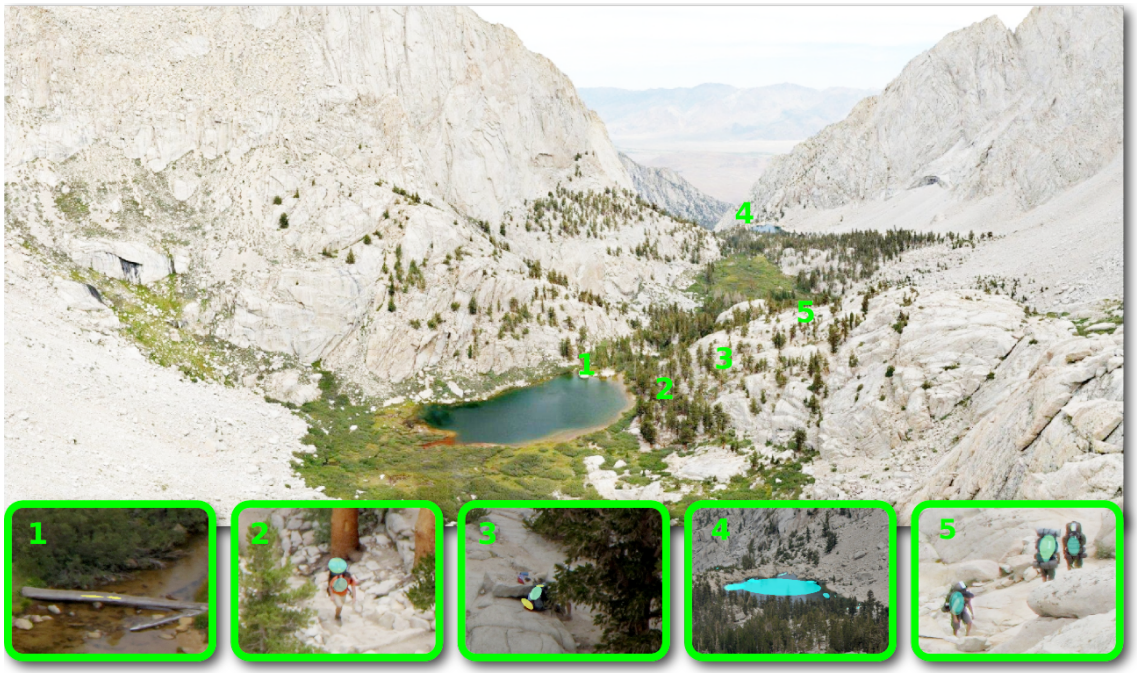


(a) Grimsel Pass (1.3 gigapixels)

(b) Royal Gorge Bridge (1.4 gigapixels)

Figure 3.11: Gigapan picture datasets with samples of numbered detected regions shown in the thumbnails. 1

(a) Cacti (4.0 gigapixels)



(b) Mt. Whitney (5.0 gigapixels)

Figure 3.12: Gigapan picture datasets with samples of numbered detected regions shown in the thumbnails. 2

landscape such as a glacier. This shows the generality of our approach.

In the Grimsel Pass picture (Figure 3.11(a)), the system detects buildings in thumbnails 1 and 8. Thumbnail 2 shows a blue Swiss Tardis. Cars, coaches, and road signs are found in thumbnails 3, 4, and 5. Thumbnail 6 shows a glacier in the mountain. Thumbnail 7 gives a view of the parking lot example in Section 3.2.

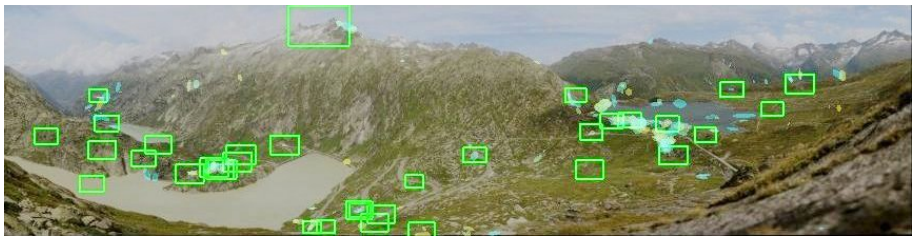
In the lower left of the Royal Gorge Bridge picture (Figure 3.11(b)), the system finds a train and a river rafting boat along the Arkansas river in thumbnails 1 and 2. We see two cable cars in thumbnails 3 and 4; the one in 3 has just departed the station whereas the one in 4 is much closer to the camera. Thumbnails 5 and 6 show a caravan and a restaurant sign around the town. Thumbnail 7 shows tourists and flags.

We found a few hikers among the cacti in Figure 3.12(a). Thumbnail 1 shows the back of a hiker; the system has identified him by his jeans. Thumbnail 2 shows a double image of two hikers, who probably moved and were captured at two instances. The hiker in thumbnail 3 was using a camera. We find a man sitting in thumbnail 4. Two other hikers are detected in thumbnail 5.

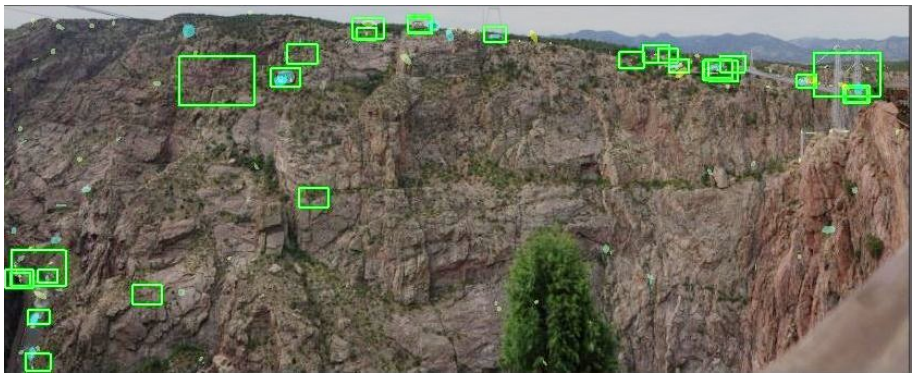
Although Mt. Whitney (Figure 3.12(b)) is a popular hiking spot, we detect more than the hikers. Thumbnail 1 and 4 show a bridge and the Alpine lake. We found 4 hikers in thumbnails 2 and 5. A hiking backpack is shown in thumbnail 3. We are able to detect 12 out of 13 hikers in this picture.

Table 3.1 shows a summary of the detected number of regions after each step. We inspect the results and count the number of detected meaningful objects, such as humans, vehicles, and buildings, after each processing step. These are shown in parentheses. Many detected regions may map to different parts of the same meaningful object.

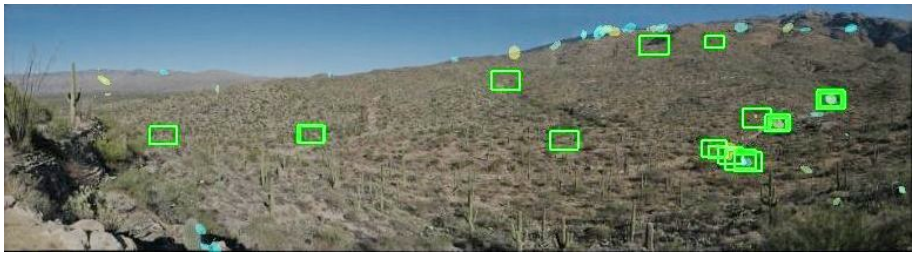
The number of detected objects remains the same until the user-guided interactive refinement step. This shows our system is conservative and generally does not remove positive results. Our system finds a few hundred regions from images of a few gigapixels.



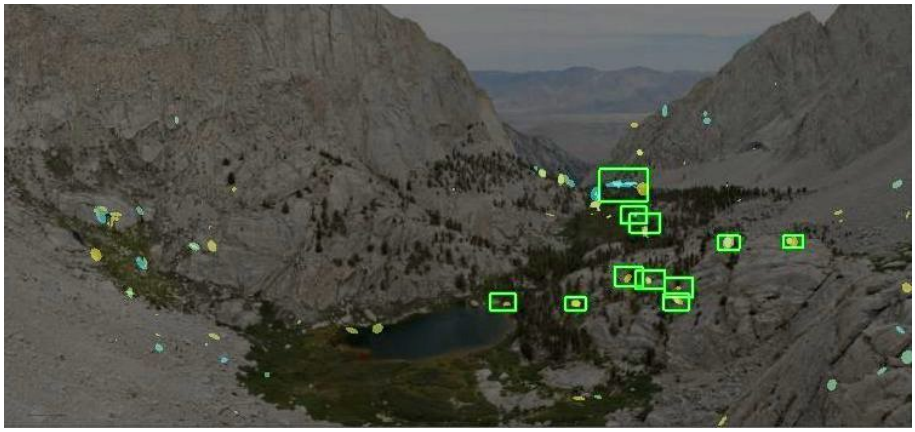
(a) Grimsel Pass



(b) Royal Gorge Bridge



(c) Cacti



(d) Mt. Whitney

Figure 3.13: This figure shows the Gigapan community tags and our detected regions. The rectangles are the Gigapan tags and the ellipses are our user-refined detected regions.

Image	SW Saliency	k-NN	Interactions
Grimsel Pass	18k (64)	525(64)	400(64), 325(62)
Royal Gorge	19k (49)	567(49)	226(49), 121(45)
Cacti	50k(10)	1.5k(10)	761(10), 513(7)
Whitney	40k(15)	1069(15)	604(14), 259(12)

Table 3.1: This table shows the quality of results after each step of processing. The SW Saliency column shows the results after sliding-window saliency map. The k-NN column shows the top 3% regions selected by anomaly detection. The Interaction column shows the results after three and five user interactions. In each of these columns, the number shows the count of computer-selected regions and the number in parenthesis is the count of the detected objects of interest. An object of interest may contain multiple detected regions.

Gigapan Community Tags

Internet users tag and comment on these very large images on Gigapan's website. Table 3.2 shows our system is able to detect a majority of the gigapan community tags. Figure 3.13 compares the tags with our user-refined results. These tags contain high-level semantic information. For example, the pattern we detected in thumbnail 1 of Figure 3.11(a) is a child's drawing for a construction accident prevention campaign. Although a large number of tags represent regions of interest, such as vehicles or humans, not all interesting regions are tagged. Figure 3.14 shows two examples of tags that contain semantics beyond general visual information. In Figure 3.14 (a), a user has tagged a common cactus as being the *original one*. Another user has tagged some buffelgrass in Figure 3.14 (b) because it grew after a fire. Clearly such tags require high-level semantic knowledge that a purely low-level perception-based system such as ours is unable to detect. We show the number of tags without semantic information in the second column of Table 3.2.

3.6.3 Performance

We report timings for different stages of our system. Table 3.3 shows the pre-processing times for the sliding-windows saliency computation, k-nearest neighbors anomaly

Image	Gigapan Tags		Detected by our system
	All	Non-Semantic	
Grimsel Pass	35	32	25
Royal Gorge	35	30	24
Cacti	24	17	15
Whitney	11	11	10

Table 3.2: This table compares our results with Gigapan community tags. Our system detects most of the Giganpan community tags. The second column shows the count of tags without semantics as discussed in Section 3.6.2. There is a difference between the count of tags and detected objects. Each tag may cover multiple objects and not all meaningful objects are tagged. Some of the tags also overlap with one another.

Figure 3.14: Gigapan community tags: (a) The original cactus. (b) Some buffelgrass after a fire. These tags contain semantics beyond general appearance.

detection, and user-guided interactive refinement.

The running time for the sliding-window saliency computation averages to 2.5 hours per gigapixel. Since the saliency computation routines are parallelizable, we expect a cluster of machines can easily process each gigapixel within a few minutes. k-nearest neighbor anomaly detection takes less than a minute. This is considerably faster than the recent image saliency techniques in Section 2.2. This pre-processing can be computed offline.

The k-nearest neighbor anomaly detection reduces thousands of salient regions to a few hundreds while it also enables the interactive refinement process. The interactive select-slide-delete refinement process (searching, and redrawing) takes only tens of milliseconds. In contrast, when we try to interactively refine thousands of salient regions, each interaction takes several seconds.

Image	Preprocessing		Interaction	
	SW Sal.	k-NN	w/o k-NN	w/ k-NN
Grimsel Pass	3.2h	5s	1193ms	9ms
Royal Gorge	4.5h	6s	1247ms	9ms
Cacti	10.3h	25s	2029ms	20ms
Whitney	11.1h	23s	1968ms	17ms

Table 3.3: This table shows the running time of our system. The preprocessing time includes the sliding-window saliency (SW Sal. column) and anomaly detection (k-NN columns). The interaction column show timings for interactive user refinements. This select-slide-delete refinement cannot be interactive without the k-nearest neighbors (k-NN) anomaly detection step.

3.7 Conclusions and Discussion

In this chapter, we show how to use visualization and computation to assist the exploration of very high-resolution large-scale landscape images. We address the visual scale challenge by introducing the sliding-window computational saliency model. Anomaly detection automatically discovers information while visualization allows the users to explore the image interactively. Our system implementation is scalable to large datasets by using out-of-core methods. We show our system can detect interesting details from various landscape images. The detections largely match community user tags.

We believe the field of visualization will be considerably strengthened by incorporating analysis and visualization of very large images as a first-class data primitive next to volumes and meshes. This chapter presents some of the first steps towards that goal.

Chapter 4

Hierarchical Exploration of Volumes using Multilevel Segmentation of the Intensity-Gradient Histograms

Identifying and visualizing meaningful features in large volume datasets remains a significant challenge [73]. Visualization of the features that you know are in the data is hard. It is even harder to find the features that you do not know are there. While we have made significant strides in building up a substantial body of knowledge over the last two decades in direct volume rendering, much of these advances have addressed issues surrounding *how* to depict the data; *what* to depict remains an important problem. We seek a visualization approach that highlights meaningful information, guides users to explore, and allows the users to associate high-level knowledge with low-level raw data. We carry this out by visually segmenting the intensity-gradient histogram of a volumetric dataset into a hierarchy for exploration.

We first outline the three most important components of the data exploration challenge facing us today.

Information Challenge: Detecting and displaying meaningful features, trends, and anamolies in data is an important challenge in visualization. This is becoming an increasingly significant challenge as our ability to acquire data is surpassing our ability to meaningfully analyze it. In this chapter we seek a way to locate and visualize data with the highest information content using simple statistically-based information-theoretic measures.

Completeness Challenge: Exhaustive data exploration is a tedious and time-intensive exercise and yet is important to ensure that we do not overlook any important features

in the data. We need mechanisms that facilitate a *complete* data exploration. In this chapter we show how to construct a exploration hierarchy to accomplish this goal using a top-down subdivision strategy to cover the entire feature space of a volume dataset.

Semantic Challenge: Current computational approaches may identify potentially informative regions by using low-level attributes such as statistics of data, the derivatives of the scalar field, or even embed the data into its own principal dimensions or manifold. However semantically driven navigation of the data is still a task that designated for the user. We facilitate addressing this challenge by providing an intuitive and interactive volume exploration interface that is based on natural groupings that are often seen to be semantically related.

In practice, users manually search for regions of interest by inspecting different areas of a feature space. Popular exploration subspaces for such a feature space include 1D density and 2D intensity-gradient. Histograms are often used to aid this search and have been implemented in several popular visualization packages, such as, Voreen [108], VisIt [19], and ImageVis3D [35]. For example, ImageVis3D provides a trapezoidal tool for this exploration task in Figure 4.1. We mimic this user search process by applying image segmentation to divide the histogram into intuitive regions at multiple scales. We show how to effectively discover regions of interest by traversing a hierarchy.

Contributions

We present a visual-data-driven approach to volume data exploration. Users explore a hierarchy to search for regions of interest from coarse to fine.

- We address the information challenge by extracting informative regions using image segmentation on reduced statistics. We mimic user explorations by visually segmenting the 2D histograms. We show these automatic 2D histogram segments well-approximate meaningful 3D volume segments. These segments fit the shape of

the histogram and cover the entire domain. We discuss this segmentation approach in Section 4.2.

- We address the completeness challenge by constructing a complete exploration hierarchy. This hierarchy organizes segments of different scales from coarse to fine. We progressively visualize the volume dataset by traversing this hierarchy. We show the construction of this hierarchy in Section 4.3.1.

- We guide the exploration by using information-theoretic measures of the volumetric data segments. We evaluate the entropies of the segments and the information gains of the subdivisions. We show how they can assist the exploration in Section 4.3.2.

- We address the semantic challenge by providing intuitive interactions to explore segments at different scales. Users can effectively identify regions of interest by traversing the hierarchy of segments. This interaction is detailed in Section 4.3.3.

Comparison

We show a visual comparison of the Tooth dataset in Figure 4.1. The corresponding segments and histograms in the intensity-gradient domain are shown beneath the rendered image. Figure 4.1(a) shows a user-specified visualization in ImageVis3D. In Figure 4.1(b), spatial transfer function [131] connects voxels and histogram pixels in a bottom-up fashion and oversegments the histogram into many regions. Our top-down segmentation is progressive and only divides the histogram into a manageable number of segments.

Region-growing techniques with parametric shapes show limited coverage. Figure 4.1(b) shows the Gaussian mixture transfer function [166, 167]. Automatic fitting produces the left visualization and requires manual resize, translate, rotate, and split operations on the transfer-function ellipses to produce the visualization on the right. Our visual segmentation approach produces a small number of freeform segments, that tightly

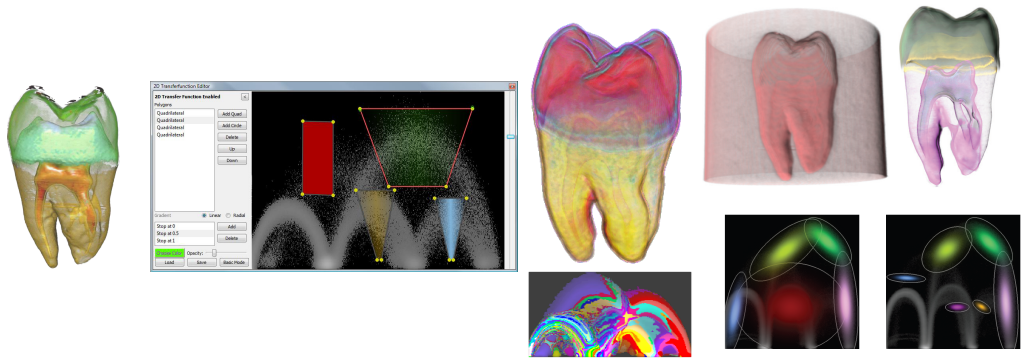


Figure 4.1: Comparison of the Tooth Dataset visualization: (a) shows a user-specified visualization in the ImageVis3D manual using trapezoidal widgets. (b) shows how the spatial transfer function oversegments the histogram into many regions; (c) shows the Gaussian mixture elliptical transfer function. The ellipses do not fully cover the histogram. The left visualization shows the automatic starting configuration. The users can then resize, rotate, and split the ellipses to obtain the right visualization. (d) shows the results from our approach in which we automatically produce a reasonable number of segments. Users can visualize the surfaces by progressively hiding the solid segments, tooth holding material, dentine, and the enamel. Images courtesy of [35, 131, 166]

fit the histogram and guarantee a complete coverage. We recursively apply the segmentation to also cover the scale space.

Figure 4.1(c) shows our results. We show the tooth surfaces by identifying and hiding solid segments of tooth holding material, dentine, and the enamel. The key advantages of our approach include:

- Assisting region search by visually segmenting the intensity-gradient histogram into collectively exhaustive and mutually exclusive segments.
- The multilevel segmentation hierarchy completely covers the dataset at all scales.
- Users interacts with a familiar and augmented feature space that is intuitive. No new features are introduced to disrupt the workflow.

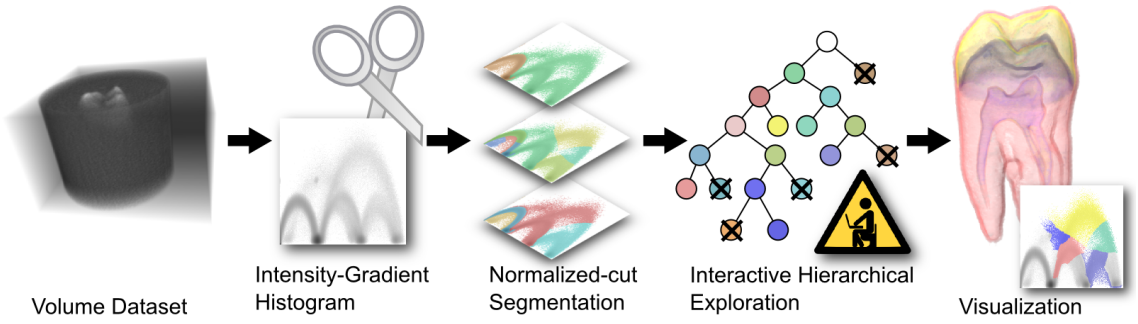


Figure 4.2: We explore a volume dataset with an intensity-gradient histogram segmentation hierarchy. 1. We compute a 2D intensity-gradient histogram from a volume dataset. 2. We mimic the user visual search of shapes in the histogram by recursively segmenting the histogram image using normalized cuts. 3. We construct a multiscale hierarchy for interactive exploration. 4. Users traverse this hierarchy to discover features in the volume data and compose meaningful visualizations.

4.1 Overview

We segment the intensity-gradient histogram of a volumetric dataset into a hierarchy of regions. We have observed that these regions express meaningful features and boundaries in the 3D volumes. The resulting hierarchy guides users to interactively explore the dataset. Figure 4.2 shows an overview of our approach.

Information Challenge: We address the information challenge by segmenting intensity-gradient 2D histograms of a volumetric dataset into potential regions of interest. The histogram is the most commonly used tool to help in transfer function design. Kniss *et al.* [87] have shown that 2D segments in the intensity-gradient domain correspond to meaningful 3D regions in the dataset. Users recognize these shapes and patterns from the histogram's image and explore the corresponding 3D regions. We mimic this user behavior by employing the normalized-cut image segmentation to extract such potential regions of interest from a 2D intensity-gradient histogram.

Completeness Challenge: The normalized-cut segments collectively span the entire intensity-gradient histogram. In order to also cover the scale space, we recursively apply

normalized-cut algorithm to obtain segments of different sizes. These segments at different scales form a multilevel hierarchy from coarse-to-fine levels of detail. We have found this to be highly usable for interactive exploration.

Semantic Challenge: To address the semantic challenge, we provide interactive exploration with the multilevel segmentation hierarchy. Users traverse through the hierarchy to sift for meaningful features. They can cull away the irrelevant segments and subdivide the relevant segments to explore the details. We evaluate the entropies and information gain in this hierarchy to aid the exploration. These information theoretic measures guide the users in deciding where to explore. The exploration results in a visualization with features at different scales and sizes.

4.2 Volume Segmentation by Normalized Cut on 2D Histograms

We aim to mimic how users would visually process a intensity-gradient histogram. Given a 3D intensity field, we compute its derivative, the gradient, to form a 2D intensity-gradient histogram. Users locate shapes and patterns from this 2D histogram to decide how to explore the 3D volume. As shown in Figure 4.1(a), users use widgets of different shapes to highlight the intensity-gradient histogram and visualize the regions of interest. We aid this tedious process by using image-segmentation algorithms to cut along the shape of this histogram. Previous work [87, 105, 131, 167] has shown that the continuity in the intensity-gradient domain reasonably approximates the spatial continuity in the dataset. We refer our readers to Kniss *et al.*'s [87] work, for specific examples of how intensity-gradient histogram shapes map to corresponding volume regions in datasets.

Normalized-cut image segmentation [144] divides the histogram into continuous shapes that we seek. It models an image as a graph and finds the best way to partition this graph into k components. Every pixel in the image is considered as a node on the graph. The edge weights, $w(u,v)$, between the nodes, u and v, are computed as color and location

similarities between the pixels. The closer the pixels, the stronger the edge weight is. The details of the similarity measure we used can be found in [25]. The normalized cut seeks to disconnect the graph, V, into components A, B by removing the edges with the least normalized cost. It therefore partitions the image along the least-similar pixels, while maintaining balanced partitions.

$$\begin{aligned}
\text{cut}(A,B) &= \sum_{u\in A, v\in B} w(u,v) \\
\text{assoc}(A,V) &= \sum_{u\in A, t\in V} w(u,t) \\
\text{Ncut}(A,B) &= \frac{\text{cut(A,B)}}{\text{assoc(A,V)}} + \frac{\text{cut(A,B)}}{\text{assoc(B,V)}}
\end{aligned}$$

where, $\text{cut}(\text{A},\text{B})$ is the total weight of edges connecting components A and B, $\text{assoc}(A,V)$ is the total weight of the edges of A in V, and $\text{Ncut}(A,B)$ normalizes $\text{cut}(A,B)$ by the edge weights of A and B. This normalization favors balanced segments, since small cuts will have a smaller denominator, and therefore, the value of $\text{Ncut}(A,B)$ is increased. Finding the minimum $\text{Ncut}(A,B)$ is a NP-complete problem. This is usually approximated by solving an eigenvalue problem:

$$\begin{aligned}
(D-W)y &= \lambda Dy \\
d(u) &= \sum_{v} w(u,v)
\end{aligned}$$

where W is the adjacency matrix of the image graph with edge weights $w(u,v)$, D is a diagonal matrix with entries, $d(u)$, and λ is an eigenvalue. We can use the resulting eigenvectors, y to partition the graph. Yu and Shi [171] show how to find k partitions by finding k eigenvectors of the eigenvalue problem.

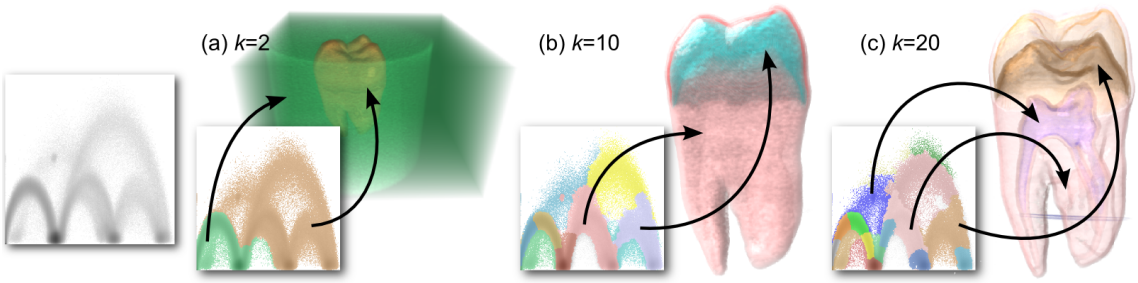


Figure 4.3: We show normalized-cut histogram segmentations of the Tooth dataset at different k's. (a) shows the separation of the tooth and the volume box. (b) shows segments of the crown and root of the tooth. At $k = 20$, (c) shows material boundaries that are separated from the solid components.

We segment the intensity-gradient histogram using a normalized-cut approach to mimic a semantically meaningful segmentation of the volume dataset. We construct a 2D histogram from the volume dataset. We compress the dynamic range of the frequencies by taking the log, such that the statistics can be represented by a grayscale image. We trade the 3D spatial connectivity information for a compact abstraction in the form of a 2D histogram. The size of the histogram representation is independent of the size of the volume; it is only dependent on the bin sizes and precision, which can be controlled during the histogram construction. The normalized-cut segmentation procedure divides a 2D histogram into multiple segments. These non-parametric segments fit the histogram tightly as they completely cover the histogram's intensity-gradient domain. In our experiments, we construct a 256×256 histogram from the dataset and store the histogram as a grayscale image. We compute the normalized cut using Cour *et al.* [25]'s software.

Figure 4.3 shows example segments of the Tooth dataset and how this normalized-cut approach to the intensity-gradient histogram segmentation results in intuitively meaningful segments. We overlay the segments onto the 2D histograms and show the corresponding 3D segments. The normalized-cut separates the tooth from exterior material at $k = 2$. At $k = 10$, the segments distinguish solid components and the tooth crown and the root are easily identified. The segments at $k = 20$, distinguish material boundaries from the solid components.

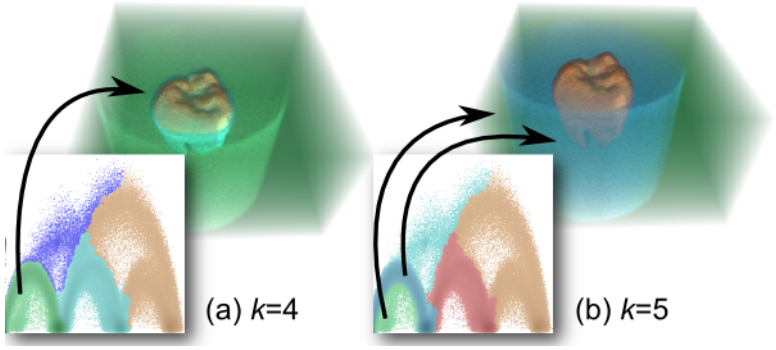


Figure 4.4: When we increase k from 4 to 5, the volume box is segmented into empty space and the cylindrical material that holds the tooth. Although this is a legitimate segmentation, it does not segment the tooth, the region of interest.

4.3 Hierarchical Exploration

We recursively segment the histogram to form a hierarchy for interactive exploration. A common problem in segmentation is determining the desired number of segments, k. Searching for an appropriate k by repeating the segmentation is a tedious exercise.

Furthermore, as k increases, the new segments may not subdivide any region of interest. For example, Figure 4.4 shows the segmentation of the Tooth dataset with $k = 4$ and $k = 5$. We see an additional segment that divides up the box instead of the tooth as k increases from 4 to 5. The newly-divided segments represent the empty space and the material that holds the tooth. Although these are legitimate segments, most users would prefer to segment the tooth.

4.3.1 Multilevel Segmentation Hierarchy

To eliminate the need for a predetermined k, we recursively apply normalized cuts to segment the histogram and build a binary hierarchy. This hierarchy guides users to explore the histogram segments from coarse to fine details. Users interactively subdivide and explore selective regions of interest. For example, the users may interactively subdivide the tooth while keeping the box intact. This segmentation hierarchy covers the entire

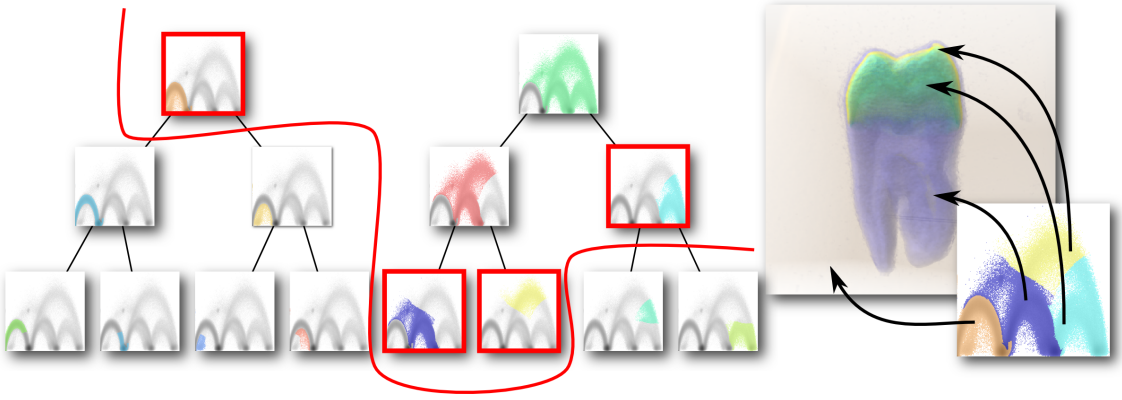


Figure 4.5: A multilevel segmentation hierarchy enables users to compose visualizations with segments of different sizes and scales. Assembling the red-framed segments produces the visualization on the right. Similar to a level-of-details hierarchy, a cut in this hierarchy covers the entire dataset.

intensity-gradient domain at all different scales to ensure an exhaustive exploration.

Any cut through this segmentation hierarchy covers the entire dataset. We can traverse this hierarchy to explore the dataset at varying levels of detail in different regions of the volume. Figure 4.5 shows how the parent histogram nodes in the hierarchy are subdivided into the children histogram nodes. In this example, we choose to explore the tooth while keeping the segment of tooth-holding materials and space intact at the first level. The second level of the hierarchy shows the segmentation of the solid tooth crown in the third arc. The third level contains segments of the tooth root and the shell of the crown. We can compose a visualization that covers the whole tooth with these segments at different scales as shown on the right of that figure.

This multilevel segmentation hierarchy provides a mechanism for users to compose visualizations by flexibly combining segments at different scales. Given the shapes of the histogram segments or corresponding volume regions, users can efficiently decide where to explore and refine. In the next section we show how information-theoretic measures can aid in the intuitive segmentation of the hierarchy.

4.3.2 Information Content

We evaluate the information content of a segment or a subdivision to aid the exploration. In addition to the user's intuition and domain knowledge, information content can serve as an auxiliary guide when traversing the segmentation hierarchy. The use of information-theoretic measures is increasingly popular in visualization. Kim and JaJa [81] build information-aware octrees to extract isosurfaces. Jänicke *et al.* [63, 65, 66] and Chen *et al.* [18] apply information theoretic measures to visualize flow. Ruiz *et al.* [135] compose automatic transfer functions based on information divergence.

We compute the entropy at each segment and evaluate the information gain at each subdivision of the segmentation hierarchy. Our goal is to guide users to explore segments that are more informative. To assist user exploration, we characterize the segments and the subdivision with two different information theoretic measures: (a) Entropy, and (b) Information Gain.

Segment Entropy: We use entropy to characterize the complexity of a segment. We extract sub-volumes, V , from the dataset according to the segments (the nodes in the hierarchy). We compute the entropy in the sub-volumes, $H(V)$, as follows:

$$H(V) = -\sum_i p(v_i) \lg_2 p(v_i)$$

where v_i is a voxel in V, $p(v_i)$ is the probability of v_i. This classic Shannon entropy measures how many bits are required to encode V. A high number of bits required represents a higher complexity.

In Figure 4.6(a), we color the segments of the Tooth dataset according to their entropy. It shows that the tooth contains a higher entropy than the empty space and tooth-holding material. However, when we further subdivide the tooth in Figure 4.6(b) the entropies of different components start converging and it becomes less clear which segment should be further explored. To address this we evaluate the information gain.

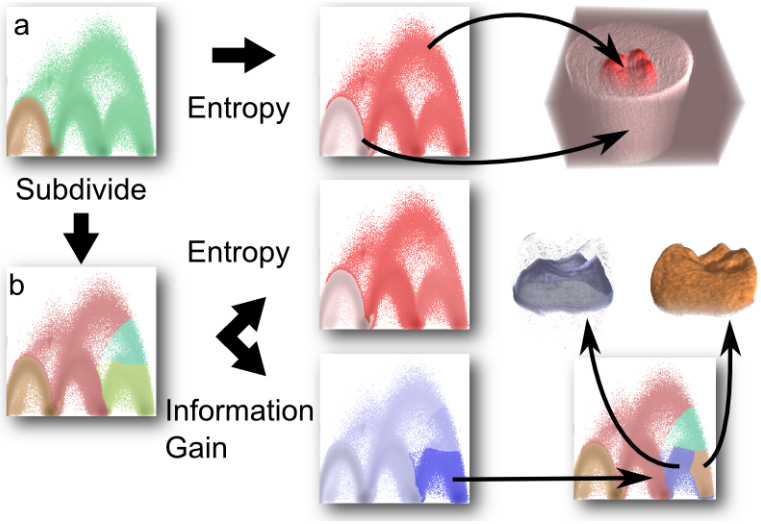


Figure 4.6: (a) shows the tooth segment has a higher entropy (in red) than the empty space and tooth-holding material. Further subdivision the tooth in (b) results in segments with converging entropies. In an alternate approach, we evaluate the information gain on those segments. The subdivision with the highest information gain (in blue) separates the dentine boundary from the enamel crown.

Information Gain: We use information gain to evaluate the effect of subdividing a segment. The information gain is the reduction in entropy after a subdivision. When the entropy reduction is low, it means that the subdivision does not sufficiently reduce the complexity of the segment. However, if the entropy reduction is high, it suggests that the complexity is lowered in the sub-segments, which is likely to happen as result of separation of two different structures into the two sub-segments. This measure has been widely used in building decision trees [129], where the attribute with the highest information gain subdivides the training data. We compute this by subtracting the entropy of the children nodes from the parent node:

$$G(V) = (H(V) - \sum_j \frac{|V_j|}{|V|} H(V_j))/H(V)$$

where V_j are the sub-volumes of V. We normalize this gain by the entropy before the subdivision. Figure 4.6(b) shows how information gain can distinguish segments with similar entropies — the dentine boundary is separated from the enamel crown.

4.3.3 Interactive Exploration

We use the segmentation hierarchy to guide volume exploration. The users interact with segments at different levels and compose visualizations on the intensity-gradient histogram. The main interactions in this procedure are subdividing or hiding a segment. Hand editing of segments is not necessary and optional. No manual translation, rotation, resizing, or reshaping of segments is required in the results presented in this work.

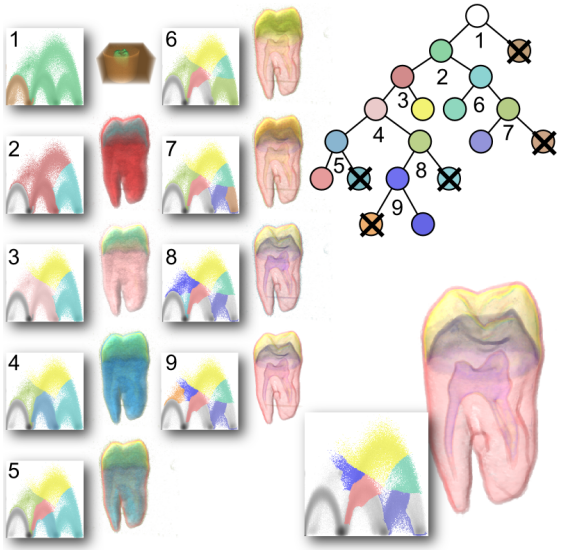


Figure 4.7: This shows an example of the interactive exploration process. The hierarchy on the right shows the subdivision steps. The left shows the corresponding visualizations and histogram segments at different steps. The process is detailed in Section 4.3.3

We present the dataset in a bi-segment configuration to start the exploration. Similar to other related work, we overlay the segments on the intensity-gradient histogram for the users to relate to the volume regions. By default, we color the segments using a randomly-generated color map. Clicking on any segment subdivides it into two components. Users can visualize the entropy or information gain using a color-coded visualization with an appropriate key press. Users may also choose to change the colors and opacities of each segment. These interactions allow a user to inspect and focus on regions of interest. We also provide shortcuts to clear the opacity of any segment to allow the user to easily cull away any segment.

Figure 4.7 presents an example of the interactive exploration process. The hierarchy on the top right shows the subdivision and the numbers indicate the order of subdivision. The corresponding histograms and visualizations are shown on the left. The colors of the nodes are the colors of the corresponding segments. The crossed-out leaves indicate that

we decided to cull-away those segments and their subtrees are not further explored.

We show how to exhaustively explore the Tooth dataset and compose a boundary visualization similar to the user-specified visualization in Figure 4.1(a). The supplementary video shows this visualization can be constructed in less than a minute. The exploration starts with the full dataset. The initial segments in step 1 divide the tooth from the rest of data, the first leftmost arc in the histogram. We focus on the tooth from step 2 onwards. Step 2 shows the crown and the root. Step 3 segments the shell of the crown in yellow. Step 4 shows the second arc is the root of the tooth and separates it from noise (this is best seen in the video). We first select the root and then remove the noise. Step 5 divides the root into the solid blob-like dentine component and its boundary. We keep the dentine boundary and move on to the tooth crown represented by the third and rightmost arc in step 6. Step 7 divides the arc, the left half shows the boundary between the enamel and the dentine and the right half shows the solid enamel component. We show the boundary by hiding the solid component and move on to explore the last noise-like segment between the first and second arcs in step 7. In step 8, the lower half of the noise-like component shows the noise, however the top half shows the boundary of the pulp along with a small piece of the tooth-holding material at the back. We hide the noise, then further segment the pulp boundary in step 9. This allows us to remove the tooth-holding material and arrive at our final visualization.

4.4 Results

We apply this exploration approach to visualize a variety of volume datasets:

1. Engine
2. Foot
3. Visible Human Male Head

4. Tomato

5. Hurricane Isabel

Implementation and Experimental Setup: We compute a six-level normalized-cut hierarchy for each histogram by using Cour *et al.*'s [25] normalized cut software in Matlab. We only store the finest level of the segmentation and access the rest according to a binary-tree order. We visualize the segments as freeform components on a 2D intensity-gradient transfer function. We implement our application with Qt and the NVIDIA CUDA SDK volume rendering example. The preprocessing in Matlab takes about 15 seconds per histogram. This preprocessing can be avoided completely if we can perform normalized cut with an accelerated eigensolver at exploration time as discussed in Section 4.5. We performed all experiments on a Linux workstation with a Intel Xeon 5140 and a NVIDIA GeForce 260GTX GPU.

For each dataset, we show a visualization and the corresponding exploration hierarchy. The numbering shows the order of the subdivision steps along the hierarchy. We illustrate the key steps and their corresponding visualizations on the left side of each figure. We also highlight segments corresponding to interesting structures of the datasets. The supplementary video shows the interactive composition process. Our results show our approach follows human intuition in segmenting histograms into meaningful components.

Engine: The engine block's histogram shows a distinctive arc-like region that represents the main engine block and the space around it. The segmentation separates this solid arc region from the other scattered arc with high intensity and gradient in the histogram. The high intensity and high gradient arc represents the internal structures and is shown in Figure 4.8(b). Figure 4.8(a) shows the arc region that represents the space and the engine block regions shows a high information gain. We subdivide the arc to separate the

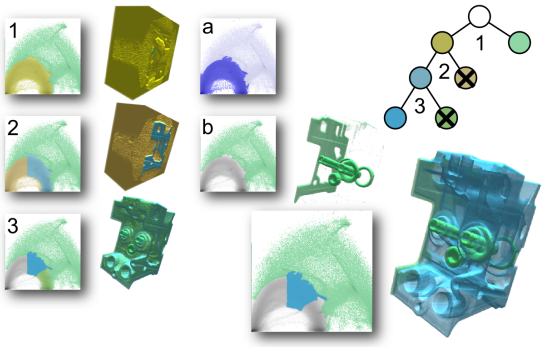


Figure 4.8: Visualization of the Engine dataset: in (a) the solid arc region shows a high information gain. We subdivide this arc to separate the engine block from the space and the noise and visualize it along with the internal structures shown in (b).

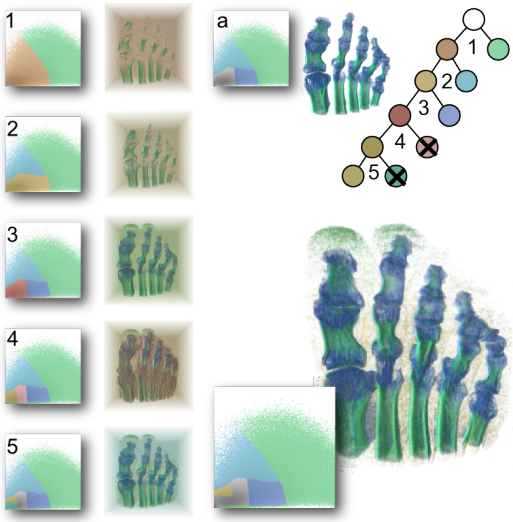


Figure 4.9: Visualization of the Foot dataset. Each component of the foot, such as, the bone (Step 1), the joints (Step 3), and the flesh (Step 4) are shown along the composition hierarchy. (a) shows the separated bones and joints.

space and noise from the engine block. We then obtain the surface of the engine block by subdividing another level. The final visualization shows the engine block surface and the internal structures in Figure 4.8.

Foot: Figure 4.9 shows the visualization of the Foot dataset. Although the histogram in this foot dataset does not exhibit any distinctive arc shapes, our visual segmentation technique can still divide the histogram into meaningful segments. Step 1 separates the bones from the rest. We subdivide the box of spaces and the flesh in steps 2-5, to obtain segments corresponding to the joints, the flesh, and the skin. Figure 4.9(b) shows the visualization of the bones along with the joints. We compose the final visualization to include the bones, the joints, and the skin.

Tomato: We distinguish different parts of a tomato in Figure 4.10. The red low intensity and high gradient region is the skin in Figure 4.10 (c). The blue locular cavities in Figure 4.10(b) are characterized by high intensity and high gradient. Notice the yellow

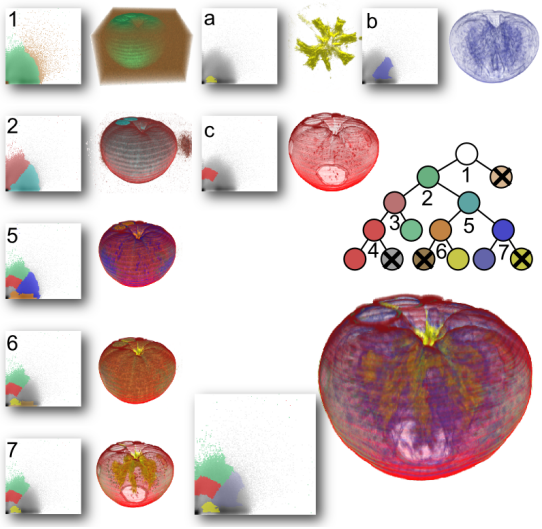


Figure 4.10: Visualization of the Tomato dataset. Steps 1-4 lead to the skin segment in (c). Steps 5-6 show the columella and placenta in (a) The locular cavity is shown in (b). The visualization shows the skin, locular cavity, columella, and placenta of the tomato.

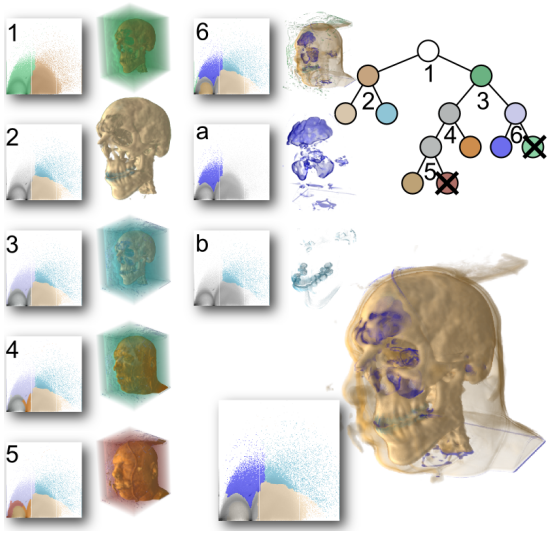


Figure 4.11: Visualization of the Visible Human Male Head dataset. The hierarchy progressively (2-6) reveals the bone, teeth, flesh, skin, and sinus. (a) shows the sinus segment and (b) shows the teeth segment. We compose the visualization with the bone, teeth, skin and sinus.

collumella, core, of the tomato in Figure 4.10(a) is a center region in the histogram, whose segmentation would not have been possible with an intensity-only 1D histogram.

Visible Human Male Head: We present the head of the visible human male dataset as an example in Figure 4.11. Similar to the Foot dataset, Step 1 shows the bones. Step 2 shows the teeth. Steps 3, 4, and 5 divide the low-intensity arc region progressively and show the flesh (step 4) and the skin (step 5). Step 6 divides the low-intensity and high-gradient regions into the sinus and some noise. We remove the noise and show the bone, skin, teeth, and sinus in the final visualization.

Hurricane Isabel: We visualize the pressure field of the hurricane Isabel dataset. This is a continuous scalar field with no abrupt boundaries and is different from the previous examples with clear material changes. Figure 4.12 shows the exploration hierarchy of the hurricane. Step 1 separates the land and the atmospheric regions. Step 2 segments the

hurricane eye from the atmospheric region. The rainbands from the eyewall structures of the hurricane are separated in Step 3. By inspecting the entropy of the segments in (c), we decide to switch our focus onto the eye of the hurricane. Steps 4 and 5 show regions of different pressures in the hurricane eye. We compose the final visualization with the hurricane eye segments, the eyewall and the rainbands. The separated eyewall and the rainbands components are shown in Figure 4.12(a) and (b).

4.5 Conclusions and Future Work

In this chapter we present a hierarchy of normalized-cut-assisted visual segmentation of an intensity-gradient histogram to assist in the volume exploration process. In contrast with existing approaches in Figure 4.1(b) and (c), our top-down segmentation approach produces fewer segments and a superior coverage. Our visual approach also well approximates user-specified visualizations in Figure 4.1(a). We address the information challenge by using a visual segmentation of intensity-gradient histograms to locate various meaningful volumetric segments. These segments completely cover the intensity-gradient domain in both the image space and the scale space. We show that any cut in the segmentation hierarchy covers the entire dataset. Exploring along this hierarchy addresses the completeness challenge. We assist the volumetric data exploration process by using information-theoretic measures. The users can identify meaning-

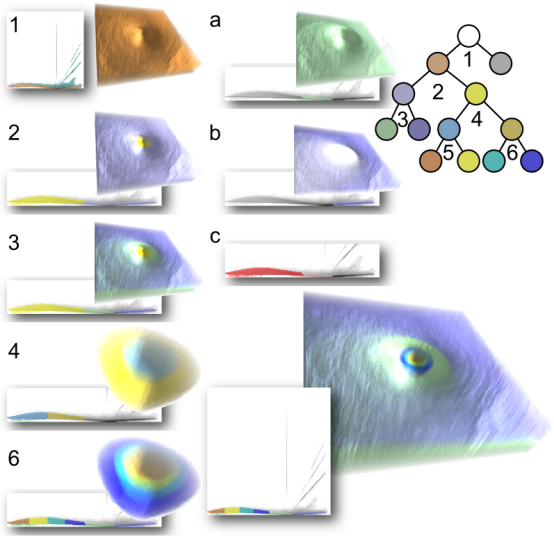


Figure 4.12: Visualization of the pressure field of the Hurricane Isabel dataset. We magnify the bottom part of the histogram in steps 2-6, (a), (b), and (c) to better show the segments. Steps 1-3 divide the hurricane into its major components: the eye, the eyewall, and the rainbands. (c) shows the entropy of the Hurricane eye is higher than the other segments after step 3. We focus on segmenting the hurricane eye in steps 4-6. (a) shows the eyewall segment and (b) shows the rainbands segment.

ful components and material boundaries through a concise interactive exploration procedure. These interactions address the semantic challenge by allowing users to adaptively explore the dataset and associate their knowledge with the corresponding data segments.

Chapter 5

Personalizing the Brain Atlas through Gene Expression Correlates

5.1 Introduction

Segmentation of biological volumes is a long-standing challenge. In this chapter we show how we can use massive data arising from gene expression profiles of hundreds to thousands of genes to complement and enhance existing methods. In contrast to previous work on using data from a single image (such as a volumetric MR image) with atlas-based segmentation of brain images, our work shows how to use correlation data from multiple gene-expression profile volumes to personalize the brain atlas.

Analysis of gene expression patterns in a three-dimensional context has revealed interesting results about the roles of specific genes in development and disease. Most studies to date have analyzed the expression of single genes within specific organs (such as heart, brain, or entire embryos) across stages of development or disease states. The spatial localization of individual genes can be determined through fluorescent *in situ* hybridization (FISH), followed by confocal microscopy, optical transmission tomography, or by imaging serial sections of (frozen or paraffin-embedded) tissue. A global view of gene expression across multiple genes can be obtained through microarray analysis of dissected tissue, either by selecting specific tissue types or through voxelization. In the latter approach, sections of immobilized tissue are segmented into spatially registered cubes (voxels) which are then subjected to microarray analyses. Voxelization is particularly appealing since the segmentation process is not biased by prior knowledge about the tissue being analyzed, making it possible to use the gene expression data to uncover novel morphological features.

Neurological disorders, such as autism, epilepsy, and schizophrenia are widely be-

lieved to have a basis in specific genes. Neurologists assess the presence or absence of candidate gene expressions in different brain regions to gain insights into these diseases [151]. While the relationship of the diseases and candidate gene expression profiles are not always well understood, it is important to facilitate this spatial-genomics exploration process through visual tools.

New medical technologies can now characterize the state of a person with increasing precision at the molecular level to the genomic level to the organ level [89]. This wealth of personal medical information is now enabling the study of customized treatment for each patient through massive data-driven knowledge discovery. Genomic data is one of the most important components in this precision and personalized medicine revolution. In this chapter we develop data-driven visual computing techniques for the analysis and visual understanding of multiple gene expression profile data to personalize the brain atlas.

Allen Mouse Brain Dataset The Allen Mouse Brain Atlas [96] provides a set spatial gene expression profiles in adult and developing mouse brains. Many gene comparison and visualization tools [152] have been published by the Allen brain institute for facilitating the use of this wealth of spatial genomic data. In addition to the gene expression profiles, it also provides an annotated reference atlas. In this chapter, we use a subset of the Allen Developing Mouse Brain dataset from the 2013 SciVis contest (`http://sciviscontest.ieeevis.org/`). Our dataset contains 1600 gene expression profiles of developing mouse brains at stage 5 (E18.5).

Contributions The main contribution of this chapter is in presenting a data-driven approach for identifying geometric features of brain structures from spatial gene expression profiles. Specifically, we show a simple pipeline for retrieving correlated genes expression profiles, extracting surfaces of the expressed regions, and personalizing a targeted reference structure.

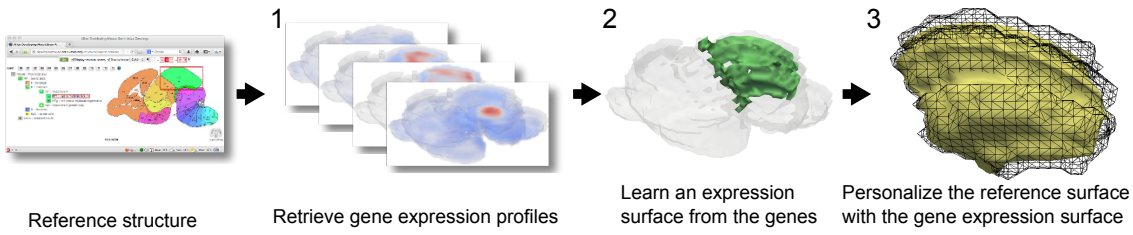


Figure 5.1: We personalize the brain atlas by learning from gene expression profiles. Given a reference structure, we first retrieve a set of relevant gene expression profiles (step 1), and then we extract an expression surface from the genes (step 2), and we finally refine the reference surface to match the extracted gene expression boundary surface (step 3).

5.2 Overview

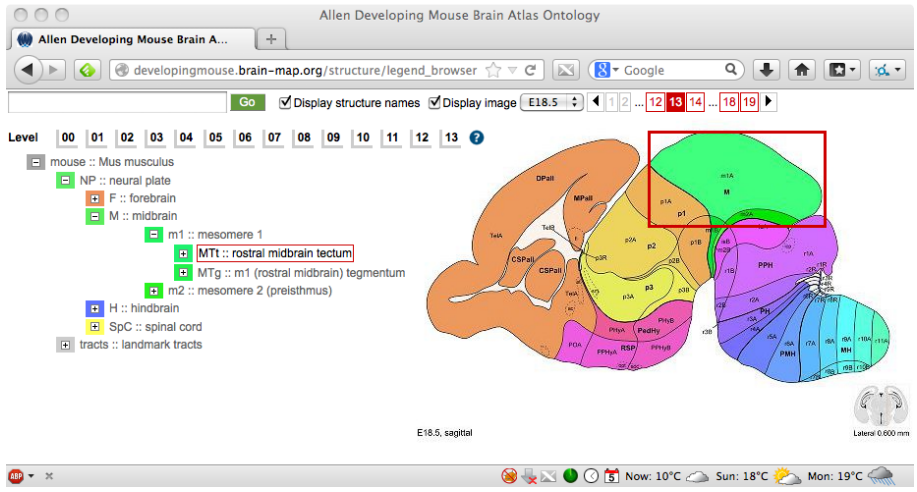
In this chapter, we extract the gene activity boundaries of a given $3D$ brain structure from a set of volumetric gene expression profiles. This data-driven procedure consists of three main steps:

1. Retrieve a set of genes that are highly correlated with the targeted reference structure.

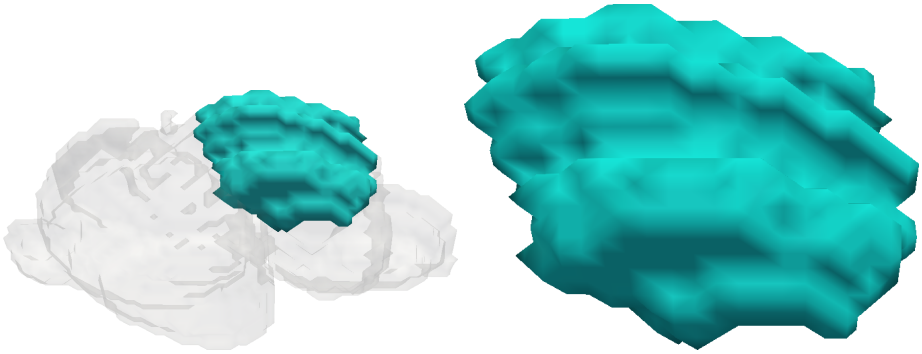
2. Construct a coherent normal vector field from the gene expression profiles, and extract a surface that represents the gene expression boundaries.

3. Personalize the surface of the reference structure to match the extracted gene expression surface.

Figure 5.1 summarizes our approach. We detail how we retrieve relevant gene expression profiles in Section 5.3. Section 5.4 describes the expression surface extraction process. Section 5.5 shows how we personalize the reference surface according to the expression surface. Section 5.6 presents our results and performance.



(a)



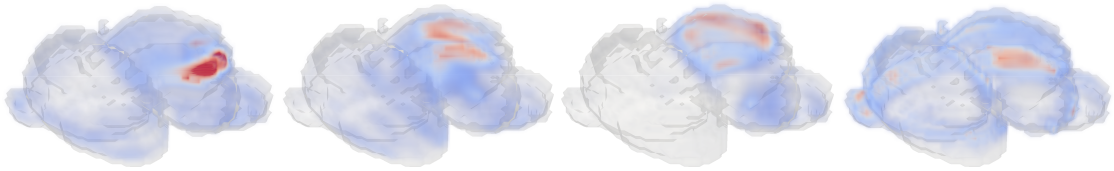
(b) (c)

Figure 5.2: Rostral midbrain tectum (MTt): (a) shows a traditional 2D annotated reference. (b) shows the 3D position of MTt in a mouse brain. (c) shows a closeup view of the surface of MTt.

5.3 Selecting Relevant Gene Expression Profiles

In order to discover high-quality and meaningful boundaries from the gene expression profiles, we need to first limit the process to a handful of the most relevant genes that are highly expressed in the target region of interest. Given a target brain structure, we first use the volumetric anatomic mouse brain atlas [96] to construct a reference binary volume mask, m. Voxels inside the target structure are labeled by ones and voxels outside the structure are labeled by zeroes. For example, Figure 5.2 shows a visualization of the rostral midbrain tectum (MTt) structure in the middle of a mouse brain.

We compute the expression level, $L(g,m)$ of a gene g and a structure mask m by



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

Figure 5.3: Volume visualization of the most expressed genes in MTt: (a) shows transcription factor gene Tfap2d. (b) shows nervous system development gene Lmo1. (c) shows transcription factor gene Tfap2b. (d) shows transcription factor gene Pou4f2. High expression energy is in red and low expression energy is in blue.

computing the dot product between the gene-expression volume and the binary structure mask volume. The gene expression profiles are volumetric scalar fields of the expressed energy. We normalize g by its total energy to ensure gene expression profiles with different energy levels are comparable. $L(g,m)$ shows the correlation of gene g with structure m.

We retrieve a set of genes, $G = g_i \cdots g_n$, with the highest $L(g,m)$ in the structure m, where $L(g,m) = \overline{g} \cdot m$ and $\overline{g} = \frac{g}{\|g\|}$. These gene energy distributions will be used to show us the boundaries of the gene activities. Figure 5.3 shows the four most active genes in the structure MTt. We retrieve five ($n = 5$) active gene expression profiles for each of our experiments.

5.4 Extracting the Segmentation Surface from Gene Expression Profiles

We extract the surface that conforms to the selected gene expression patterns. We use the highly expressed genes for this surface extraction process to ensure they are consistent with the targeted structure.

First, we construct a coherent vector field from the scalar fields of the gene expression profiles. Then, we use this vector field to derive a gradient field of gene expressions. High-gradient regions in scalar fields characterize boundaries [86, 120] between regions. Our goal is to extract the surface that passes through the high-gradient gene expression regions.

5.4.1 Coherent Normal Vector Field

We use a voting approach to identify coherent normal directions from the highly expressed genes. Note that it is insufficient to identify high gradient regions of each gene expression profile and then aggregate them, because the gradients may have different orientations and represent different surfaces. Normal vectors or tensors have been used to "vote" in robust curvature computation [155, 119] and surface reconstruction [154].

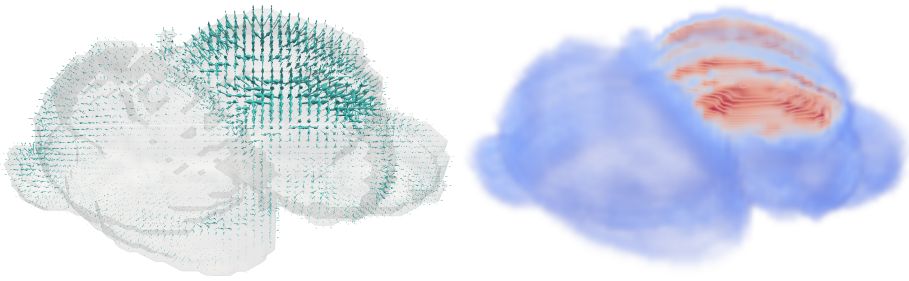
We construct a vector field of normal directions from each gene expression profile and then vote with the normals to determine the coherent directions. We construct the normals by computing the gradient of the gene-expression scalar field, g_i, along the x, y, z directions. The voting process sums the normal vectors at each voxel and produces an aggregated coherent vector field, $\vec{N}$. The magnitudes of the normal vectors are high in coherent regions and low in the incoherent regions.

$$\vec{N} = \left(\sum_i \frac{\partial \overline{g}_i}{\partial x}, \sum_i \frac{\partial \overline{g}_i}{\partial y}, \sum_i \frac{\partial \overline{g}_i}{\partial z} \right)$$

Let us consider the vector and scalar components of the coherent vector field $\vec{N}$. Let $\vec{n}$ be a unit vector field representing the directions of the vector field $\vec{N}$. Let s be the scalar field of the corresponding gradient magnitudes. We use $\vec{n}$ and s to extract an extremal surface [154] in the next step. Figure 5.4 shows visualizations of coherent vector field $\vec{N}$ and the scalar field $s = \|\vec{N}\|$

5.4.2 Extracting the Bounding Surface of the Gene Expressions

We find the gene expression boundaries by extracting the surface that passes through the gene expression gradient peaks. This surface can be extracted by the extremal surface method with the vector field, $\vec{n}$, and the gradient field, s. A point is on the extremal surface if its strength, s, is locally extremal along the direction of the normal $\vec{n} = \frac{\vec{N}}{\|\vec{N}\|}$, i.e. $\frac{ds}{d\vec{n}} = 0$.



(a) (b)

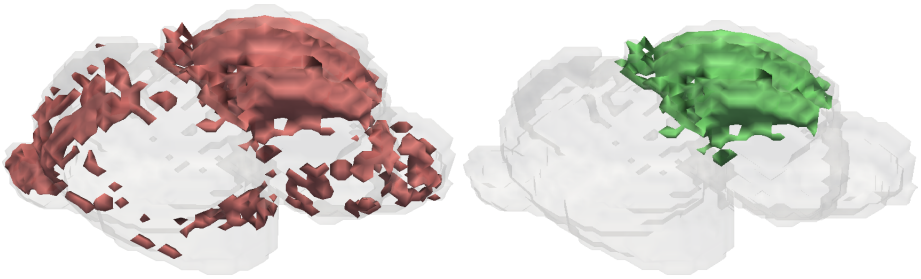
Figure 5.4: Visualization of $\vec{N}$ and s. (a) shows the coherent normal directions of the gene expression profiles in Figure 5.3. The normal vectors surrounding the target structure MTt are highly coherent. (b) shows the corresponding gradient magnitudes in colors. High gradients in red surround the target structure MTt.

Therefore, we project the vectors of ∇s onto $\vec{n}$ to obtain a scalar field q for extracting these extremal points.

$$q = \left(\frac{\partial s}{\partial x} n_x + \frac{\partial s}{\partial y} n_y + \frac{\partial s}{\partial z} n_z \right)$$

The scalar field q is the second derivative of the gene expression profile scalar fields. Extracting the extremal surface is similar to edge detection from zero-crossings of Laplacian filtered 2D images. A surface along the gradient peaks is defined at $q = 0$. We apply an iso-surface algorithm, such as Marching Cubes, to extract the 3D extremal surface from q with iso-value = 0.

We avoid extracting weak surfaces by thresholding the scalar field q, $|q| < 0.0001 = 0$. This removes the zero crossings with low gradients. Figure 5.5 (a) shows an extremal surface that we extracted from the gene expression profiles. In addition to our region of interest around the targeted MTt structure, there are many other minor co-expressed regions. To remove these regions, we identify the surface component that is the closest to the center of our targeted structure. Figure 5.5 (b) shows the surface of the component for the MTt structure.



(a) (b)

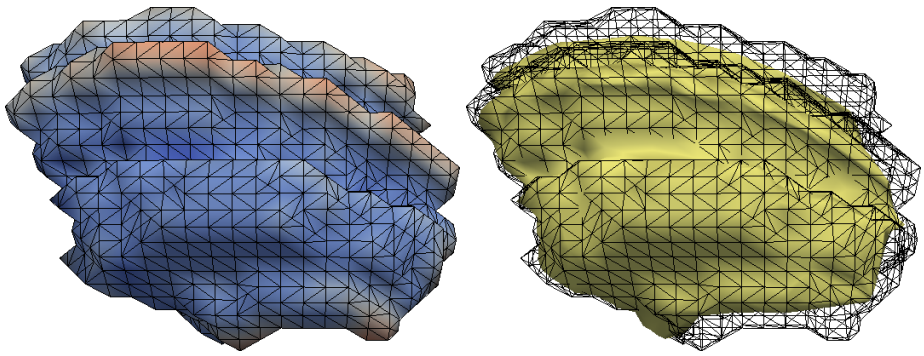
Figure 5.5: Gene expression surfaces: (a) shows the complete gene expression surface that we extracted from the gene expression profiles. There are minor co-expressed components surrounding our target structure MTt. We identify and retain the component that is the closest to MTt in (b).

5.5 Personalizing the Reference Structure to the Gene Expression Boundary Surface

We personalize the reference structure with the gene expression boundary surface by deforming the reference structure to match the latter. Given a binary reference structure volume mask m, we first extract an iso-surface at iso-value 0.5 to represent the reference structure.

For each vertex on the reference surface, we find the corresponding closest point on the gene expression surface. Many distance field computation algorithms with spatial data structures [6] and GPU computing [30, 146, 147] can perform an accelerated proximity search for this. Figure 5.6(a) shows a visualization of the distance distribution between the reference surface and the gene expression boundary surface.

We warp the reference surface to the gene expression surface by moving the reference vertices to their closest corresponding points on the gene expression surface. Figure 5.6(b) shows a comparison of the original and personalized reference surfaces.



(a) (b)

Figure 5.6: Personalizing the reference surface with the gene expression boundary surface: (a) shows the distance distribution from the reference surface to the gene expression boundary surface (large distances are shown in red and short distances are shown in blue), (b) shows a comparison of reference surfaces before (in wireframe) and after the personalization (in solid yellow surface).

5.6 Results

In this section, we present the results of our approach. For each example brain structure, we show the reference structure, the relevant gene expression profiles, the gene expression surface and the surface of our personalized reference structure. We overlay the wireframes of the reference structures to show the deformation. We first show examples with reference structures that are similar to the gene expression surfaces, and then discuss cases in which the gene expression surfaces deviate from the reference brain atlas structures.

5.6.1 Experimental Setup

We have implemented our approach on the Linux platform with Python. The normal voting and gradient computation have been implemented with `NumPy`. Iso-surface extraction and distances from surfaces are computated using VTK. We have run our experiments on a workstation with an Intel Xeon 2.33 GHz CPU and an NVIDIA GeForce GTX 295 GPU. The computation time for retrieving correlated genes, extracting the gene expression surface, and personalizing the reference structure is less than two seconds.

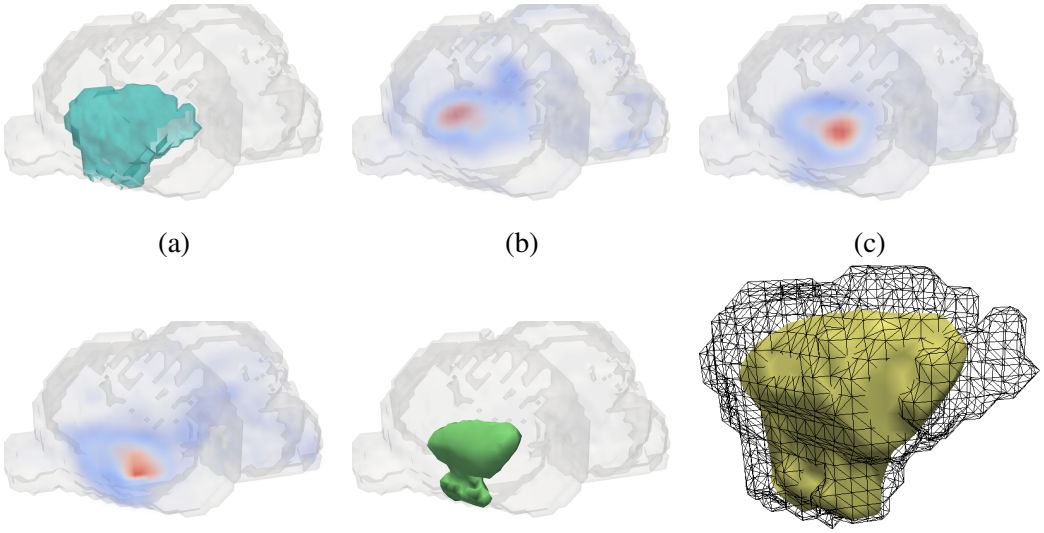
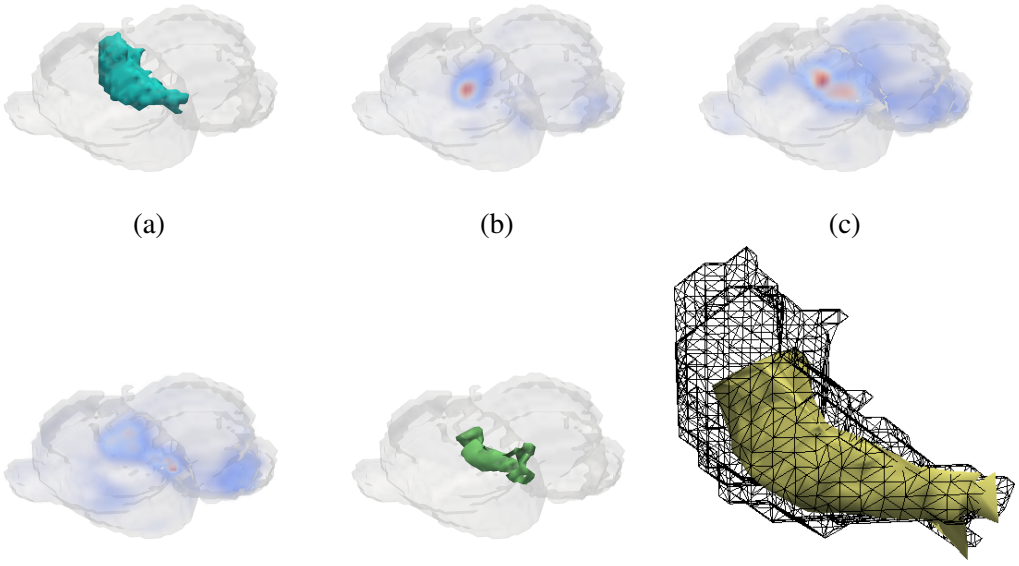


Figure 5.7: Central Subpallium: (a) shows the surface of the reference structure. (b-d) show active genes Foxp1, Pde10a, and Drd2. (e) shows the gene expression surface. (f) shows the personalized surface of the reference structure in yellow with the original reference surface in wireframe

5.6.2 Structures whose Gene Expression Surfaces largely agree with the Atlas

Central Subpallium The subpallium is the major basal subdivision of the embryonic telencephalon; it is related to the control of motor, cognitive and emotional responses. Figure 5.7(a) shows this structure. Figures 5.7 (b-e) show the relevant highly expressed genes. Figure 5.7 (e) shows the gene expression surface and Figure 5.7 (f) shows the personalized surface of the reference structure. In this example, the expressions of the three genes, Foxp1, Pde10a, and Drd2, span different parts of the targeted structure.

Alar plate of Prosomere 2 The Alar plate of Prosomere 2 is a neural structure in the developing mouse forebrain. The prosomere is a part of the thalamus, which is responsible for sensory perception and regulation of motor functions. Figure 5.8(a) shows this structure. Figures 5.8 (b-e) show the relevant highly expressed genes. Figure 5.8 (e) shows the gene expression surface and Figure 5.8 (f) shows the personalized surface of



(a) (b) (c)

(d) (e) (f)

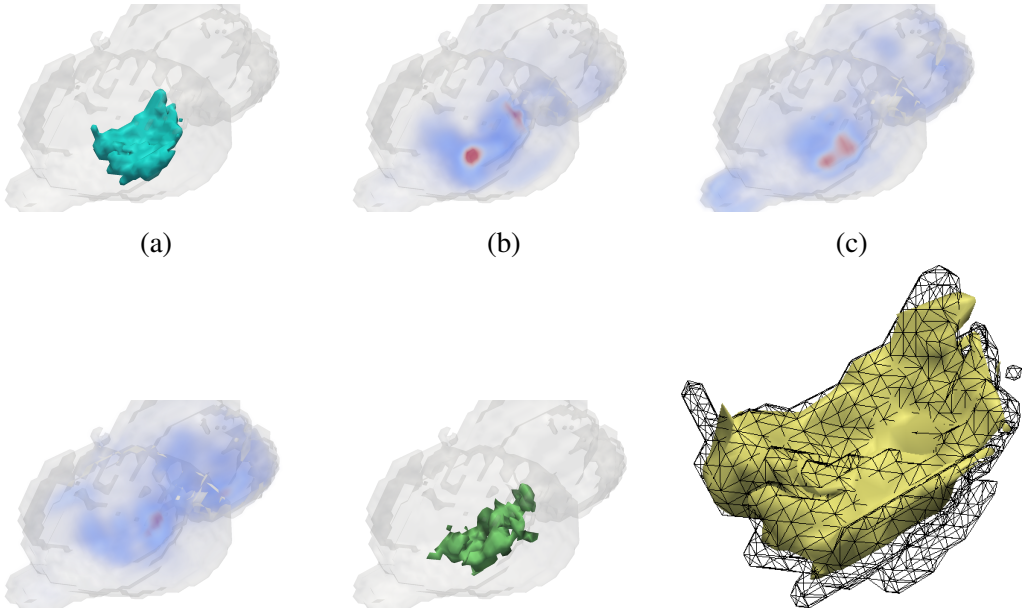
Figure 5.8: Alar plate of P2: (a) shows the surface of the reference structure. (b-d) show active genes Prox1, Ntng1, and Gbx. (e) shows the gene expression surface. (f) shows the personalized surface of the reference structure in yellow with the original reference surface in wireframe

the reference structure. The expression of gene Prox1 is highly localized. Genes Ntng1 and Gbx2 show more tube like expression patterns.

Rostral Secondary Prosencephalon This mouse forebrain development region contains the preoptic area and the rostral hypothalamus, which controls much of a mouse's basic behavior, including feeding. Figure 5.9(a) shows this structure. Figures 5.9 (b-e) show the relevant highly expressed genes. Figure 5.9 (e) shows the gene expression surface and Figure 5.9 (f) shows the personalized surface of the reference structure. The personalization deforms the reference surface unevenly according to expressions of the three genes.

5.6.3 Structures with Small Gene Expression Regions

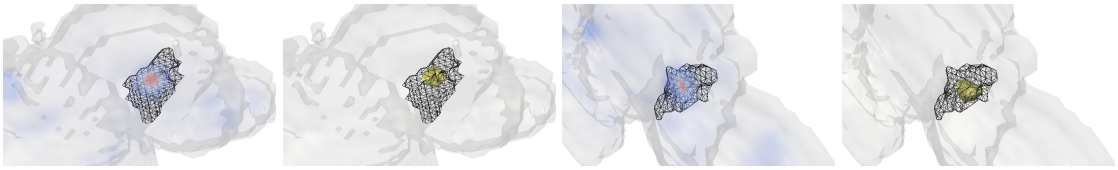
We have found structures with limited gene expressed regions whose gene expression surfaces are not similar to the surface of the reference structure. In Figure 5.10(a)



(a) (b) (c)

(d) (e) (f)

Figure 5.9: Rostral Secondary Prosencephalon:(a) shows the surface of the reference structure. (b-d) show active genes Nkx2-1, Trh, and Dlk1. (e) shows the gene expression surface. (f) shows the personalized surface of the reference structure surface in yellow with the original reference surface in wireframe



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

Figure 5.10: Some of the genes are only expressed in a small region within a structure. We overlay the wireframe reference structure on the gene expression profiles to visualize the difference. (a) shows the expression of gene Pclo in structure (b) r1 basal plate and (c) shows the expression of gene Gcg in (d) r2 basal plate .

and (c), we show the gene expression profiles for r1 and r2 basel plates. These genes are highly expressed only around the center of these plates. Our approach extracts small gene expression surfaces in Figure 5.10 (b) and (d).

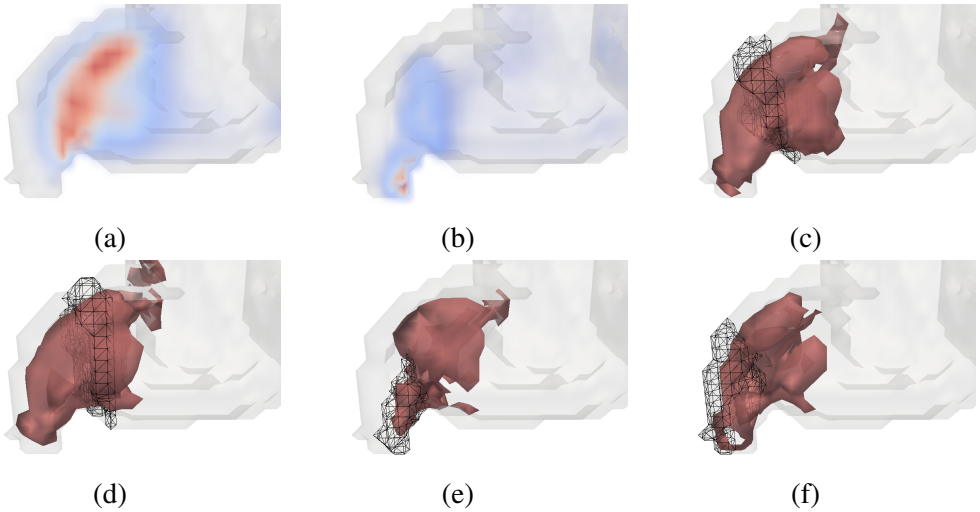


Figure 5.11: Some of the genes are expressed across multiple structures. (a-b) show example genes that are spread in a large hindbrain region. In this case, we extract gene expression surfaces that span across multiple structures.

5.6.4 High Gene Expressions in Multiple Structures

In contrast with the previous scenario, we have also found structures with high gene expressions across multiple structures. For example in the hindbrain of a mouse, the experts have identified 11 structures. We found that many gene expression profiles span across multiple hindbrain structures as shown in Figure 5.11 (a) and (b). In this case, when we extract gene expression surfaces for each of the individual structures, we get surfaces that span across multiple hindbrain structures as in Figure 5.11 (c-f).

5.7 Conclusions and Discussions

We have presented a data-driven approach for personalizing the brain atlas with many gene expression profiles. We first retrieve the relevant genes for a reference brain structure, then we extract a coherent expression boundary from these genes and use this surface of the expression boundary to personalize the reference brain structure. In the surface extraction process, we perform normal voting to identify gene expression directions that are coherent across multiple gene expression profiles. From these directions,

we extract an extremal surface for personalizing the reference brain structure. In our experiments, we have found some gene expression surfaces that are similar and some that are different from the reference brain structures. Our surface extraction approach shows the localization of spread of these gene expressions and visualizes their comparison with the reference structures. We believe our approach suggests a way in which data from multiple gene expression profiles could augment brain atlases, and in fact atlases of the entire organisms.

Chapter 6

Fast Trajectory Clustering using the Kernel Distance

6.1 Introduction

The study of time-varying geospatial patterns is important for many disciplines. The study of such patterns can greatly enhance our understanding of evolving spatial interactions and relationships amongst moving entities at varying scales of space and time. The advances and availability of tracking devices, such as RFID tags, GPS signals, as well as tracking from aerial and satellite imagery has led to an explosive growth of location-tagged data of people [175], animals [169], vehicles [127, 175], and weather data [114]. For example, climate scientists study historical hurricane patterns to help predict future hurricane tracks. The study of animal movements reveal their migratory behavior and can help in protecting endangered species from poaching. Understanding vehicular traffic patterns can lead to more informed urban planning. While we are now able to track the movements of billions of entities every second, the sheer size of this data has far surpassed our ability to meaningfully analyze it. It is crucial for us to develop highly efficient algorithms if we are to make sense out of this enormous and growing body of geospatial trajectory data. In this chapter we present a novel approach to compare and cluster these geospatial trajectories.

6.1.1 Challenges

The fundamental challenge in trajectory clustering is to efficiently compute the distance between two trajectories. This traditionally has been a quadratic operation in the number of points of the two trajectories as we have to compute the distances between all pairs of points between the two trajectories. For example, the computation of the Fréchet

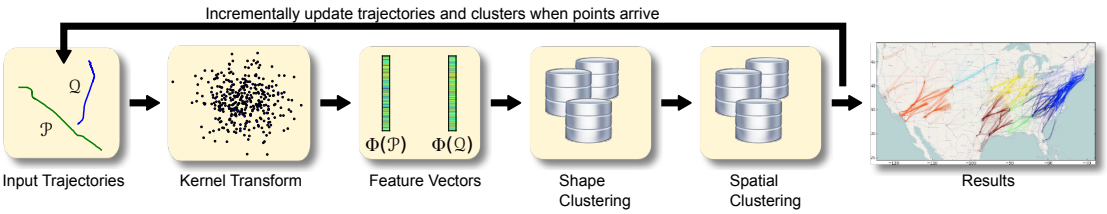


Figure 6.1: Overview: The kernel distance framework transform trajectories $\mathcal{P}$ and $\mathcal{Q}$ to fixed-length feature vectors $\Phi(\mathcal{P})$ and $\Phi(\mathcal{Q})$. We cluster the feature vectors to find similar shaped trajectories and then we cluster the trajectories according to their locations.

distance, which measures the minimum distance between the points in two trajectories, requires an $O(n^2)$ solution for polygonal curves [1] and discrete points [29].

In this chapter, we develop a new framework to compare geospatial trajectories around *approximate kernel distances* [139, 75]. In our approach we use the approximate kernel distance to transform the set of points in a trajectory to a single high-dimensional feature vector through a randomized mapping procedure. We compare the shapes of the trajectories by the distance between their respective high-dimensional embeddings.

6.1.2 Contributions

1. We introduce the use of the kernel distance for clustering geospatial trajectories. We show that the formulation of the approximate kernel distance allows us to compactly represent trajectories as high-dimensional feature vectors. This transformation allows us to leverage a wealth of point-based clustering algorithms for analyzing geospatial trajectories.

2. We use the kernel distance framework and k-means clustering to perform effective shape and spatial clustering. We also show how to update the trajectory feature vectors to accommodate streaming trajectory data. While we use the popular k-means algorithm for our clustering, our framework allows us to use any clustering method. An advantage of the kernel distance approach is that along with the distance, we also get a measure of similarity between trajectories.

3. Our approximate kernel distance framework for trajectory clustering is fairly flexible. In addition to *k*-means clustering, we also show how to incorporate automatic community detection in our framework. Such graph-based clustering approaches, facilitate discovery of clusters that are of general shapes and distributions.

4. We show that our approach is significantly more efficient than the state-of-the-art approaches in analyzing geospatial trajectories. Our approach only took about one minute to process $24K$ air traffic trajectories of 2 million points, while TRACLUS [95] took more than 5.5 hours. Our approach is also scalable and we were able to incrementally process almost half a million taxi trajectories collectively comprising 11 million points in 15 minutes. Our datasets are significantly larger than those reported in the previous work.

6.2 Overview

In this chapter, we introduce an approach to cluster geospatial trajectories by using the kernel distance. Figure 6.1 illustrates the steps of our approach. Given trajectories input $\mathcal{P}$ and $\mathcal{Q}$, the kernel distance framework projects each trajectory onto a randomly drawn i.i.d. sample of points from a normal distribution, and obtain fixed-length feature vectors $\Phi(\mathcal{P})$ and $\Phi(\mathcal{Q})$. We cluster the trajectories by clustering their feature vectors. We first compare and cluster the feature vectors according to their shape. Then we take a small uniform sample of points from each of the similarly-shaped trajectories to perform clustering according to their spatial locations. The resulting clusters contain similarly shaped trajectories that are located together.

We present the details of our approach in the following sections. Section 6.3 reviews the theories and the process of transforming trajectories into feature vectors. Section 6.4 shows how to cluster the trajectories with the *k*-means algorithm. Section 6.5 shows our shape and spatial clustering procedure, and Section 6.6 describes how to update the fea-

ture vectors for incremental clustering in a streaming fashion. Section 6.7 show results of clustering trajectories with k-means. In addition to k-means, we show how to use community detection algorithms with the approximate kernel distance framework to address some deficiencies of k-means. Section 6.9 show results of clustering trajectories with community detection. We present our performance and compare our results with existing techniques in the literature in Section 6.10. We conclude with discussions and future work in Section 6.11.

6.3 Approximate Kernel Distance

In this section, we define kernel distance and describe a well-known procedure to approximate it efficiently. In the context of this chapter, a *kernel* is symmetric similarity measure $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ between pairs of points in $\mathbb{R}^d$. Nearby points have a larger similarity value than points further away. One popular kernel is the Gaussian kernel, $K(p,q) \propto e^{-\|p-q\|^2/2}$. Given two point sets $\mathcal{P}$ and $\mathcal{Q}$ with n and m points, respectively, we define the similarity function $\kappa(\mathcal{P},\mathcal{Q})$ and compute the *kernel distance* $D_k(\mathcal{P},\mathcal{Q})$ as follows:

$$\kappa(\mathcal{P},\mathcal{Q}) = \frac{1}{mn}\sum_{p\in\mathcal{P}}\sum_{q\in\mathcal{Q}} K(p,q) \tag{6.1}$$

$$D_k(\mathcal{P},\mathcal{Q}) = \kappa(\mathcal{P},\mathcal{P}) + \kappa(\mathcal{Q},\mathcal{Q}) - 2\kappa(\mathcal{P},\mathcal{Q}) \tag{6.2}$$

The kernel distance is a metric when the similarity kernel is positive definite (like the Gaussian kernel). It is well known [139] that for any positive definite kernel, $K(p,q)$, there exists a function $\phi(p)$ mapping the point p to a Hilbert space $\mathcal{H}$ such that $K(p,q) = \langle \phi(p), \phi(q) \rangle_{\mathcal{H}}$ where $\langle .\ , .\rangle_{\mathcal{H}}$ refers to the inner product in this space. This Hilbert space is often known as a *feature space* and the function $\phi(p)$ as a *feature map*. This feature map-

ping offers a number of advantages. The linearity of dot products allows us to represent nonlinear shapes, such as trajectories, as a *single feature vector*, $\Phi(\mathcal{P}) = \sum_{p \in \mathcal{P}} \phi(p)$, and the kernel distance $D_k(\mathcal{P}, \mathcal{Q}) = \| \Phi(\mathcal{P}) - \Phi(\mathcal{Q}) \|_{\mathcal{H}}$ (the Euclidean distance). In essence, the embedding linearizes the metric by mapping the input (nonlinear) space into a vector space. Therefore, kernel methods have been widely used in the machine learning literature to recognize non-linear structures in data.

The feature map, described above, is not readily usable for computational purposes since the feature space $\mathcal{H}$ is usually infinite dimensional. However, it is often possible to construct a ρ-dimensional feature map that approximates the kernel sufficiently well. Rahimi and Recht [130] address this problem for shift-invariant kernels (where $K(p,q) = k(p-q)$). Their *randomized Fourier feature map* is based on Bochner's theorem which states that for properly scaled shift-invariant kernel, its Fourier transform $g(\omega)$ is a proper probability distribution. If $\phi_\omega(p) = e^{j\omega^T p}$, Rahimi and Recht [130] showed that $\phi_\omega(p)\phi_\omega(q)^*$ is an unbiased estimator of $k(p-q)$ when ω is drawn from g. In particular,

$$K(p,q) = \int_{\mathbb{R}^d} g(\omega) e^{j\langle \omega, p-q \rangle} d\omega = E_\omega[\langle \phi_\omega(p), \phi_\omega(q) \rangle] \tag{6.3}$$

Their mapping is described in Procedure 1.

Procedure 1 Rahimi and Recht's [130] randomized Fourier feature map

Input: Positive definite shift-invariant kernel $K(p,q)$, ρ
Output: Randomized feature map $\tilde{\phi}(p) : \mathbb{R}^d \to \mathbb{R}^\rho$

Compute Fourier transform $g(\omega) = \frac{1}{2\pi} \int e^{-j\omega^T \delta} k(\delta) d\delta$
Draw $\rho/2$ iid samples $\omega_1, \omega_2, \ldots, \omega_{\rho/2} \in \mathbb{R}^d$ from g
Let $\tilde{\phi}(p) \equiv \sqrt{\frac{2}{\rho}} [\cos(\omega_1^T p), \ldots, \cos(\omega_{\rho/2}^T p),$
$\sin(\omega_1^T p), \ldots, \sin(\omega_{\rho/2}^T p)]^T$

Joshi *et al.* [75] show that to achieve an ε-approximation of the kernel with proba-

bility $\geq 1-\delta$, it is sufficient to use $\rho = O(1/\varepsilon^2 \log(n/\delta))$ dimensional embedding. Since ρ has a logarithmic dependence on n, the approximate kernel distance computation takes $O(n \log n)$ time instead of the quadratic complexity of the exact method.

In this chapter, we use the Gaussian kernel in all our computations. Since it is a shift-invariant kernel, we can use the randomized Fourier feature approach to perform the embedding. For this kernel, we sample the ω vector from a normal distribution.

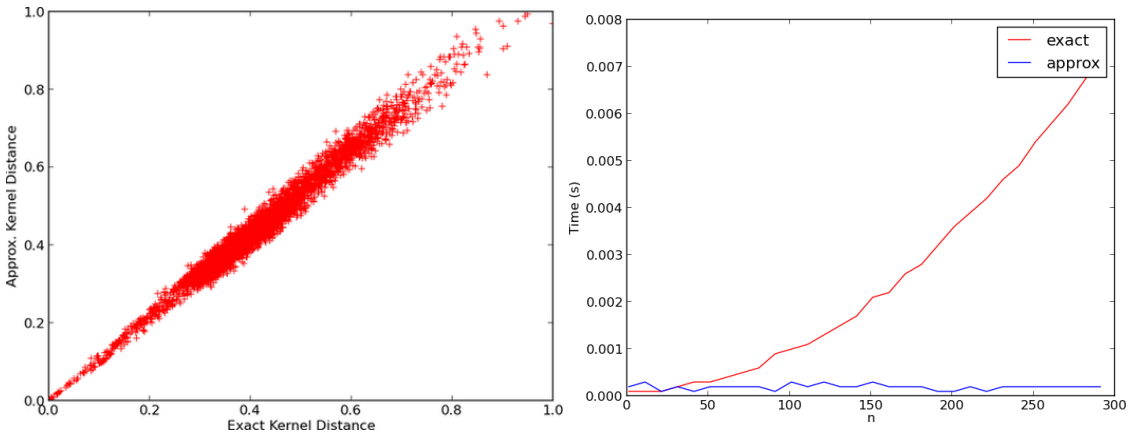
6.3.1 Clustering Trajectories with the Kernel Distance

We can cluster trajectories with the kernel distance by clustering their feature vectors with existing Euclidean clustering algorithms. $\kappa(\mathcal{P}, \mathcal{Q})$ can be computed efficiently by a dot product of the approximated feature vectors of $\tilde{\Phi}(\mathcal{P})$ and $\tilde{\Phi}(\mathcal{Q})$, where $\tilde{\Phi}(\mathcal{P}) = \sum_{p \in \mathcal{P}} \tilde{\phi}(p)$,

$$\kappa(\mathcal{P}, \mathcal{Q}) = \sum_{p \in \mathcal{P}} \sum_{q \in \mathcal{Q}} \langle \tilde{\phi}(p), \tilde{\phi}(q) \rangle = \langle \sum_{p \in \mathcal{P}} \tilde{\phi}(p), \sum_{q \in \mathcal{Q}} \tilde{\phi}(q) \rangle = \langle \tilde{\Phi}(\mathcal{P}), \tilde{\Phi}(\mathcal{Q}) \rangle$$

In light of the above identity, it is not difficult to see that the Euclidean distance of feature vectors, $\tilde{\Phi}(\mathcal{P})$ and $\tilde{\Phi}(\mathcal{Q})$, yields the kernel distance between the two trajectories [75]. We can perform kernel-distance-based clustering of trajectories by using a wealth of Euclidean clustering algorithms. Furthermore, these feature vectors only have to be computed once for many repeated distance calculations. This is extremely efficient for clustering applications.

We model the continuity of the trajectories by adding the velocity to the trajectory points. We compute the discrete differences of points (dx, dy) in a trajectory and represent each point by a 4D vector (x, y, dx, dy), where x, y represent the point location, and dx, dy represent the velocity. Note that the dimensionality of embedding, ρ, or the running time of the kernel distance approximation does not depend on the dimension of the trajectory points. The extra attributes do not incur any overhead after the kernel transform operation.



(a) (b)

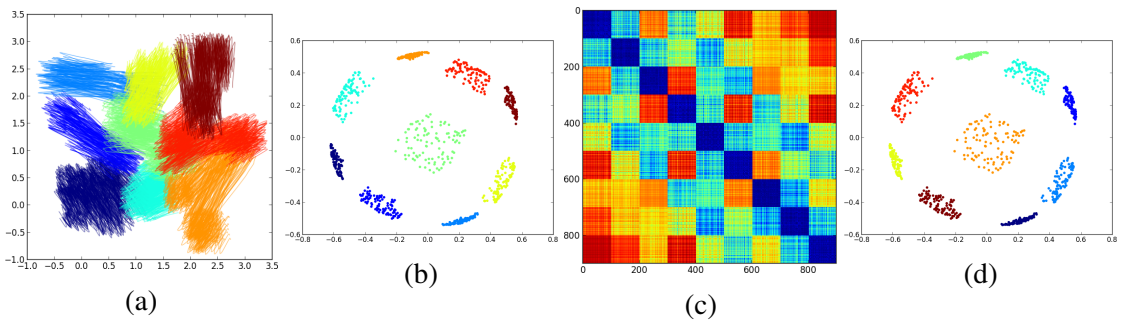
Figure 6.2: (a) shows the error distribution of the approximate kernel distance ($\varepsilon = 0.2$, $\delta = 0.2$) in comparing 100 random trajectories. (b) shows a comparison of timings for computing the exact kernel distance (red) and the approximate kernel distance (blue).

6.3.2 Evaluating the Approximate Kernel Distance

In our approach we use the kernel distance that compares the trajectories by computing a Gaussian-weighted sum of differences between all pairs of points in the two trajectories. The exact kernel distance is computed as in Equation (2) with the Gaussian kernel. For the approximate kernel distance computation we use $\varepsilon = 0.2, \delta = 0.2$ for all the experiments in this chapter. Figure 6.2 shows the error distribution and the performance of the approximate kernel distance versus the exact kernel distance.

The error plot, shown in Figure 6.2(a), is generated by comparing 100 real air traffic trajectories sampled from our dataset. Notice that the error of the approximate kernel distances is very low for similar trajectories, which is important for building clusters. The error grows only as the distance increases. The root mean square error is about 5% showing that the quality of the approximation is high.

The performance plot, Figure 6.2(b), is generated by comparing random trajectories of size n as shown on the x-axis. The running time for computing the exact kernel distance scales up quadratically, while the running time for computing the approximate kernel distance is almost linear.



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

Figure 6.3: Clustering of Synthetic Trajectories with k-means: (a) shows the trajectories. (b) shows the PCA plot of the trajectories. (c) shows the matrix of approximate kernel distance between the trajectories. (d) shows of k-means clustering discovers the correct clustering.

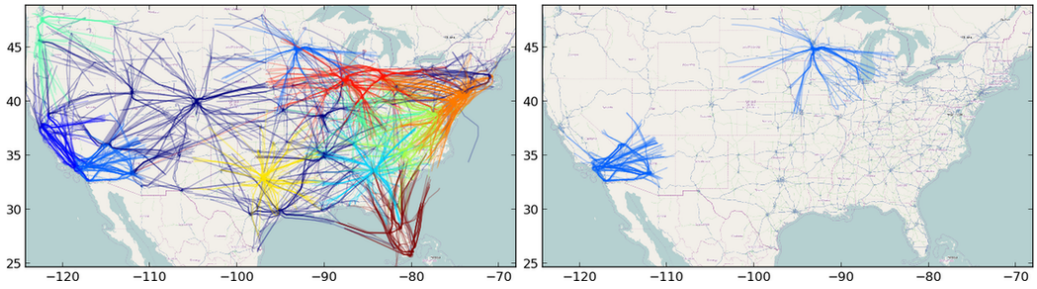
6.4 Clustering Feature Vectors with k-means

This approximate kernel distance approach can be directly used with a member of off-the-shelf clustering algorithms. We use synthetic examples to show the distribution of the high-dimensional feature points. We first generate clusters of synthetic trajectories, project them onto the high-dimensional feature space, and then cluster them.

We use the midpoint displacement algorithm from fractal generation [109] to create trajectories of different shapes. For each trajectory, we rotate and translate it to create clusters of minor variations. These varying trajectories form clusters for our experiments. Figure 6.3(a) shows an example of these trajectories.

We project these trajectories onto a set of random basis as described in Section 6.3. In Figure 6.3(b), we visualize the distribution of the high-dimensional feature points by projecting them onto the first two principal components from the principal component analysis (PCA).

In Figure 6.3(c), we visualize the approximate kernel distance among the synthetic trajectories by a distance matrix, where low distances are in blue and high distances are in red. The matrix is sorted in the order of the clusters; the distances should ideally be low around the diagonal (same cluster) and high around off diagonal regions (different clusters).



(a) (b)

Figure 6.4: We cluster 2000 flight trajectories into 10 clusters by using the kernel distance and off-the-shelf clustering techniques. (a) shows k-means Clustering. (b) shows one of the k-means clusters contains two disjointed components. We will show how to obtain more refined clusters in Section 6.5

Datasets with good separations such as the presented example can be effectively clustered by a simple clustering algorithm, such as the k-means. Results of k-means clustering is shown in Figure 6.3(d).

Figure 6.4 shows k-means clustering of 2000 air traffic trajectories with the kernel distance approach. It identifies clusters around the hub airports. However, some of the clusters produced by this naive k-means approach contain disconnected components on this real dataset. We next discuss how to address this issue by shape and spatial clustering of the trajectories.

6.5 Shape and Spatial Clustering of Trajectories

We introduce a two-step procedure to cluster the shapes and the locations of the trajectories. The shape step leverages the shape matching abilities of the kernel distance framework and the spatial step ensures the disjointed clusters are separated. We construct and cluster the shape and spatial feature vectors of the trajectories as in Procedure 2.

T is a set containing N trajectories $\mathcal{P}_1, \ldots, \mathcal{P}_N$. s and l are the desired numbers of shape and spatial clusters. C is a set of cluster labels for each trajectory. V_S and V_L are the shape and spatial feature vectors for clustering. For notational ease, we assume there are n points in each trajectory.

Procedure 2 Shape and Spatial Clustering

Input: Trajectories $T = \{\mathcal{P}_1, \ldots, \mathcal{P}_N\}$
Number of Shape Clusters s
Number of Spatial Clusters l
Output: Cluster Labels $C = \{C_1, \ldots, C_N\}$

$V_S \leftarrow \{\}$
for all $\mathcal{P} \in T$ **do**
$\mathcal{P}' = \mathcal{P} - \frac{1}{n}\sum_{p \in \mathcal{P}} p$
$V_S \leftarrow V_S \cup \{\tilde{\Phi}(\mathcal{P}')\}$ ▷ **Shape Feature Vector**
end for
$C \leftarrow k\text{-means}(V_S, s)$ ▷ **Shape Clustering**
for $i = 1$ to s **do** ▷ **Process the Shape Clusters**
$V_L \leftarrow \{\}$
for all $C_j = i$ **do**
$V_L \leftarrow V_L \oplus \{\text{Subsample}(\mathcal{P}_j)\} \oplus \frac{1}{n}\sum_{p \in \mathcal{P}_j} p$
▷ **Spatial Feature Vector**
end for
$\{C_j \mid C_j = i \wedge C_j \in C\} \leftarrow k\text{-means}(V_L, l) + (i-1) \times l$
▷ **Spatial Clustering**
end for

We first remove the spatial information from the trajectories by subtracting their centers of mass. We use the kernel transform to construct shape feature vectors, V_S. These vectors are clustered into s clusters by using k-means clustering.

In the second step, we construct spatial feature vectors, V_L, to locate the trajectories for spatial clustering. One simple spatial feature of a trajectory is its center of mass. In order to make the spatial feature vector more discriminating, we uniformly sample a few points from each trajectory. The points are concatenated together (denoted by $\oplus$ operator in Procedure 2) to form a spatial feature vector for clustering. These spatial feature vectors ensure that only spatially proximal trajectories are clustered together. We compute the spatial feature vector by sampling the center of mass and four extra points from each of the trajectories. We perform k-means clustering (with $k = l$) to obtain l spatially coherent clusters from each of the s similarly-shaped trajectory clusters. This procedure produces $s \times l$ clusters of trajectories with coherent shapes and locations. Parameters s and l control the granularity of the shape and spatial (or location) clustering, respectively.

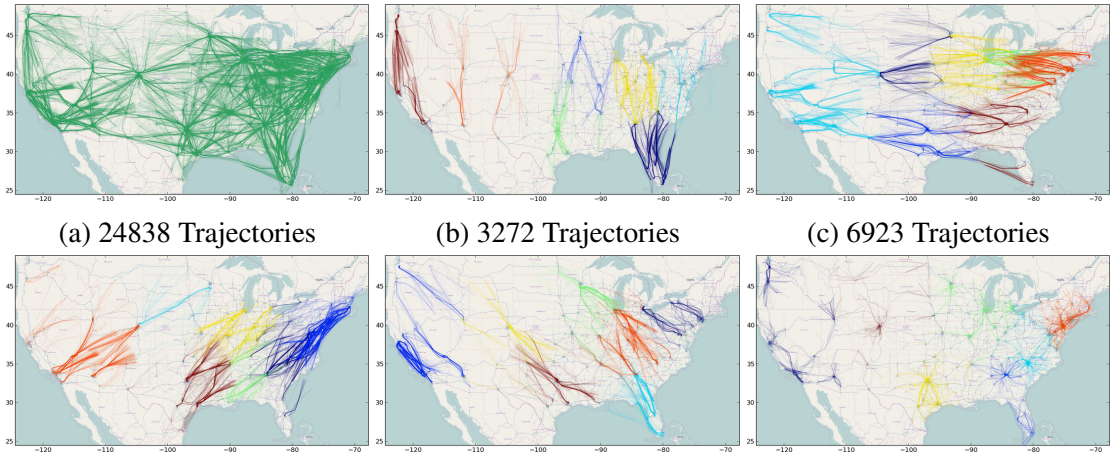
6.6 Incremental and Stream Processing

Clustering trajectories with the kernel distance can be performed in an incremental and/or streaming fashion. The summation nature of the feature vectors $\tilde{\Phi}(\mathcal{P})$ allow us to update them incrementally. In addition, since there is no need to store the raw trajectory locations after updating the feature vectors, this approach is also suitable for stream processing. Incremental and stream processing of trajectories is highly desirable for geospatial applications because new locations are continuously reported as tracked entities such as people, animals, and vehicles move.

We update the feature vectors and clusters as discussed in Procedure 3. $\mathfrak{P} = \{\mathsf{P}_1 \cdots \mathsf{P}_N\}$ is the newly arrived set of trajectory points at timestep t. V^{t-1} and C^{t-1} are the sets of feature vectors and cluster labels from the previous timestep $t-1$. We compute a feature vector, $\tilde{\Phi}(\mathcal{P}^t)$, for trajectory $\mathcal{P}$ at time t by computing a weighted sum of the feature vector of the new points, $\tilde{\Phi}(\mathsf{P})$, and the feature vector of the trajectory at $t-1$, $\tilde{\Phi}(\mathcal{P}^{t-1})$. We assign the weights, w^t and w^{t-1}, in proportion to the number of points represented by $\tilde{\Phi}(\mathsf{P})$ and $\tilde{\Phi}(\mathcal{P}^{t-1})$. If there is any new trajectory, we assign it to its closest cluster. We seed the k-means clustering procedure with cluster assignments of the previous timestep to accelerate convergence.

Procedure 3 Incremental Trajectory Clustering

Input: New Points $\mathfrak{P} = \{\mathsf{P}_1 \cdots \mathsf{P}_N\}$
Feature Vectors $V^{t-1} = \{\tilde{\Phi}(\mathcal{P}_1^{t-1}) \cdots \tilde{\Phi}(\mathcal{P}_N^{t-1})\}$
Cluster Labels $C^{t-1} = \{C_1 \cdots C_N\}$
Weights w^{t-1}, w^t
k Clusters
Output: V^t, C^t
for all $\mathsf{P} \in \mathfrak{P}$ **do**
$\tilde{\Phi}(\mathcal{P}^t) = w^t \tilde{\Phi}(\mathsf{P}) + w^{t-1} \tilde{\Phi}(\mathcal{P}^{t-1})$
$V^t \leftarrow V^t \cup \{\tilde{\Phi}(\mathcal{P}^t)\}$ ▷ **Update the feature vector**
end for
$C^t \leftarrow k$-means(V^t, k) w/initial labels C^{t-1}



(a) 24838 Trajectories (b) 3272 Trajectories (c) 6923 Trajectories

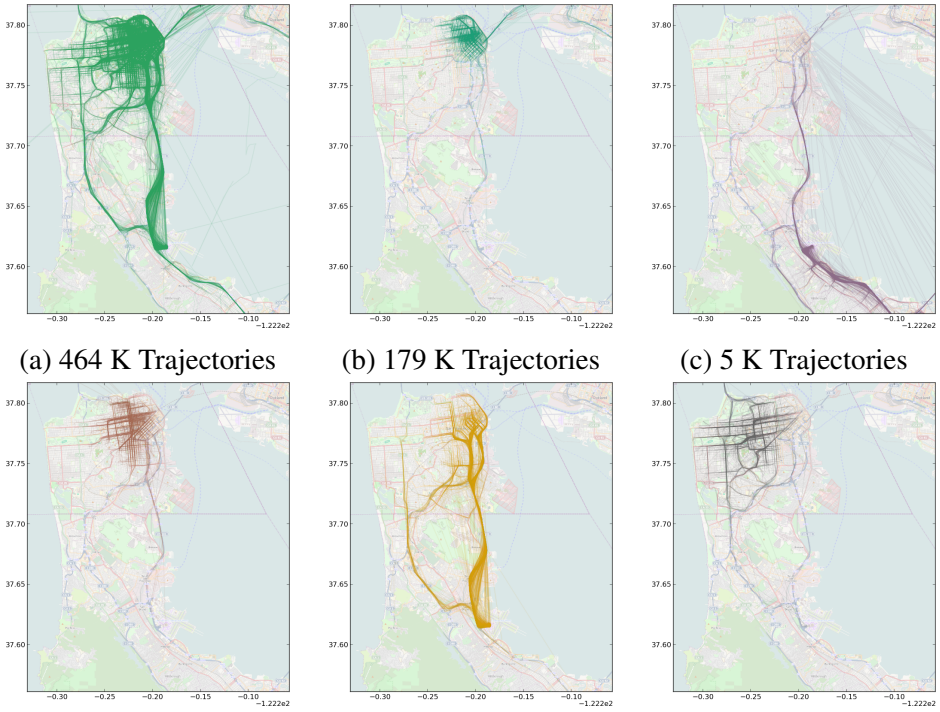
(d) 6295 Trajectories (e) 3820 Trajectories (f) 4528 Trajectories

Figure 6.5: **US Air Traffic**: (a) shows all the $24K$ flight trajectories. (b) to (f) show the five shape clusters. The colors indicate the spatial cluster assignment to different flights produced by our algorithm. (b) to (e) show flight trajectory clusters of different orientations. (b) shows flights from North to South. (c) shows flights from East to West. (d) shows flights from Northeast to Southwest. (e) shows flights from Southeast to Northwest. Interestingly, (f) shows our approach has found groups of short flights in star-shaped clusters from popular airport hubs, including Seattle and San Francisco in the West, Denver in the North, Dallas, Atlanta, and Miami in the South, and a series of airport hubs from DC to Boston in the East.

6.7 Results with k-means Clustering

We have validated our approach by using two very large datasets (1) air-traffic over the continental United States, and (2) vehicular traffic in an urban environment. Our application datasets of air-traffic ($2M$ points) and taxis ($11M$ points) are significantly larger than the data used in the previous work ($100K$ points [98]) on trajectory clustering.

Experimental Platform We have performed all our experiments on a PC with an Intel Xeon 5140 2.5 GHz processor and 4 GB of memory. Our software was implemented on the Linux platform with Python and the Numpy numerical package for computing the kernel transform and the Pycluster [27] package for performing k-means clustering.



(a) 464 K Trajectories (b) 179 K Trajectories (c) 5 K Trajectories

(d) 183 K Trajectories (e) 50 K Trajectories (f) 45 K Trajectories

Figure 6.6: **San Francisco Taxi**: (a) shows all of the 464 K taxi trajectories. (b) to (f) show the top clusters produced by our algorithm. (b) shows traffic in Northeastern San Francisco with connections to Oakland and Berkeley. (c) shows the traffic that passes the 101 highway with a cluster at San Mateo. The line pattern in (c) is due to faulty GPS data, however it does not affect our results, this shows our approach is reasonably robust to noise. (d) shows the traffic in downtown and Southern San Francisco. (e) shows a popular loop that connects the downtown to the San Francisco Airport. (f) shows the traffic in Western San Francisco. We adjust the transparency of the trajectories to visualize their relative densities in each figure, since (a) is dominated by clusters (b) and (d), some detailed patterns in the small cluster (c) may not be visible in (a).

6.7.1 US Air Traffic Dataset

The GE FlightQuest [41] dataset consists of a whole day of airplane flight trajectories over the United States airspace. There are a total of $24,843$ flight tracks with well over two million GPS points. Clustering air traffic data helps us to understand the airspace usage and see patterns among flight routes. This is important for air traffic planners and authorities to improve the flight schedules, plan new routes, develop plans for avoiding congestion, and to keep air transportation flowing efficiently.

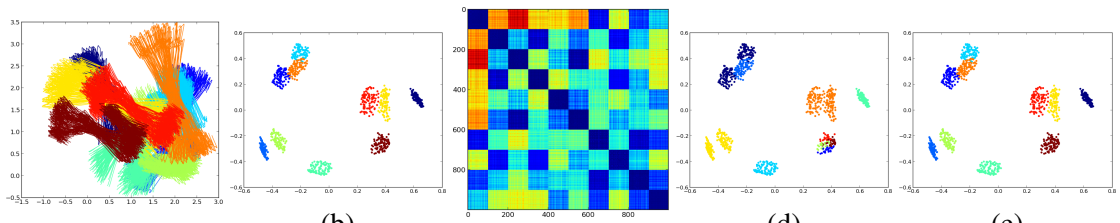
We use our shape and spatial clustering approach to cluster the flight trajectories

with $s = 5$ and $l = 7$. We first cluster the flight trajectories into different orientations then cluster them according to their locations. Figure 6.5(a) shows all $24K$ flight trajectories. Figures 6.5(b)-(f) show the five shape clusters. The colors show the spatial clusters of the flights. As can be clearly seen, four of the five shape clusters separate North to South flight trajectories (Figure 6.5(b)), East to West flights (Figure 6.5(c)), Northeast to Southwest flights (Figure 6.5(d)), and Southeast to Northwest flights (Figure 6.5(e)). Figure 6.5(f) interestingly groups short flights in star-shaped clusters from popular hub airports, including Seattle and San Francisco in the West, Denver in the North, Dallas, Atlanta, and Miami in the South, and a host of airports from Washington, DC to Boston in the East.

6.7.2 San Francisco Taxi Dataset

This dataset consists of a few weeks of taxi trajectories in San Francisco. There are a total of $464,000$ trajectories and 11 million GPS locations. These taxi trajectories represent taxi routes while under hire by a passenger. Understanding traffic patterns is important for urban planning and road maintenance. Since the taxis can move freely in an urban environment, their trajectories are very different from the planned straight flight paths and the smoothly-curved hurricane tracks. The high variations amongst trajectories makes this a very challenging dataset. The taxi data is provided by `cabspotting.org` and has been collected by EPFL for studying mobile wireless networks [127]. It is available from the CRAWDAD [91] data archive.

We cluster this taxi dataset by using our incremental clustering approach with the kernel distance. We use the k-means algorithm to directly cluster the feature vectors of the taxi trajectories into five clusters. Figure 6.6(a) shows all $424K$ taxi trajectories. Figure 6.6(b) shows a cluster of Northeast San Francisco with connections to Oakland and Berkeley. Figure 6.6 (c) shows the 101 highway and a cluster at San Mateo. Figure 6.6(d) shows a cluster of traffic at Central and South San Francisco. Figure 6.6(e) shows a



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

Figure 6.7: Clustering of synthetic trajectories with community detection: (a) shows the trajectories. (b) shows the PCA plot of the trajectories. (c) shows the matrix of approximate kernel distance between the trajectories. (d) shows k-means cannot discover the correct clustering. (e) shows the multilevel community detection discovers the correct clustering.

popular loop that connects the San Francisco downtown to the San Francisco Airport. Figure 6.6(f) shows a cluster in West San Francisco. This example shows our approach can meaningfully separate the city and suburban traffic of different shapes.

We compute the clustering of taxi trajectories by using the incremental update strategy. For each increment we loaded 4 million new points and constructed trajectories for clustering. We use the cluster assignments before the increment to seed the clustering process. Our performance is summarized in Section 6.10.

6.8 Clustering Feature Vectors with Community Detection

While we have shown k-means shape and spatial clustering works well with the approximate kernel distance framework. k-means clustering is not without limitations. It is well known that k-means only works well in discovering evenly distributed spherical clusters of similar sizes. Furthermore the number of k clusters is sometimes not know in advance.

Figure 6.7 shows a synthetic example that k-means does not handle well. Note that the clusters are not distributed as evenly as the example in Figure 6.3 and they are of more heterogeneous shapes. Figure 6.7 (a) shows the trajectories. Figure 6.7(b) shows the PCA plot with points colored according to the clusters. Figure 6.7(c) shows a visualization of

the distance matrix, the off-diagonal blue region shows members of different clusters may be similar. Figure 6.7(d) shows how the k-means approach fails to identify the clusters.

We apply community detection [36] with the approximate kernel distance framework to address these deficiencies. Many community detection algorithms extract connected structures from graphs without the need for specifying the number of components. They seek connected clusters within graphs that maximize a certain quality metric, such as the graph modularity. Graph modularity [113] measures the unexpected fraction of the edges that fall within the given clusters minus the expected fraction of (random) edges. Since the community detection algorithms only operate on the edge strengths, they also make no assumption of the clusters' shapes.

We compute a k-nearest-neighbors graph of the high-dimensional feature points to approximate the data geometry. The edge weights between the vertices representing trajectories $\mathcal{P}$ and $\mathcal{Q}$ are modeled by the kernel $e^{-\|\tilde{\Phi}(\mathcal{P}),\tilde{\Phi}(\mathcal{Q})\|}$.

Figure 6.7(e) shows clusters detected by the multilevel community detection algorithm [11]. This is a fast bottom up algorithm that attempts to maximize the modularity by moving graph vertices between the communities iteratively. It runs in almost linear time for sparse graphs.

6.9 Results with Community Detection

We detect different groups of Atlantic Ocean hurricane trajectories by combining approximate kernel distance and multilevel community detection. We use an updated Atlantic Hurricane dataset which consists of more data than the previous study [95]. Note that this dataset consists of more overlapping curves from different clusters than the air traffic and taxi datasets.

6.9.1 Atlantic Hurricane Dataset

This dataset consists of the tracks of the Atlantic Ocean hurricanes. There are a total of 1476 trajectories with $42,203$ points. Clustering hurricane trajectories is important for studying hurricane formation and for predicting the path of new hurricanes. This dataset is provided by the NOAA National Hurricane Center [114].

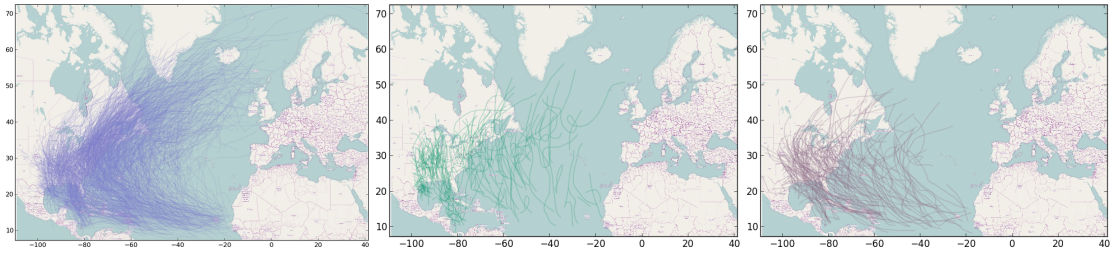
We replace the k-means clustering algorithm in our approach with the multilevel community detection [11] algorithm. For each clustering step, we construct a k-nearest-neighbor ($k = 50$) graph of the trajectories with the approximate kernel distance.

The multilevel community detection algorithm automatically detects five clusters from the hurricane trajectories. Figure 6.8(a) shows all the 1476 hurricane tracks. Figures 6.8(b)-(f) show the five automatically detected communities. Figure 6.8(b) shows hurricanes flowing from South to North and without much curvature. Figure 6.8 (c) and (d) show hurricanes that weaken after landfall in the North East of the US. Figure 6.8 (c) shows hurricanes that flow from the South Atlantic Ocean and Figure 6.8 (d) shows hurricanes that flow from the Mexican Gulf. Figure 6.8 (e) shows the strong "C" shape hurricanes that flow from the South Atlantic Ocean, hit the US North East, then exit back to the Atlantic Ocean. Figure 6.8 (f) shows southern hurricanes that weaken immediately after the landfall. These results show that our approach can separate trajectories into intuitive and meaningful clusters.

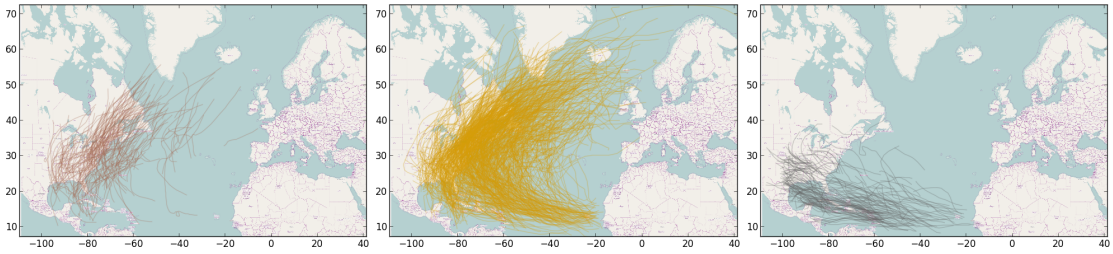
6.10 Comparison and Performance

6.10.1 Comparison with TRACLUS

We compare the performance and resulting clusters of our kernel distance clustering approach against the popular TRACLUS software provided by Lee *et al.* [95]. We used the provided programs to estimate the optimal parameters for TRACLUS clustering and performed experiments on the Hurricane dataset and the US Air Traffic dataset. We report

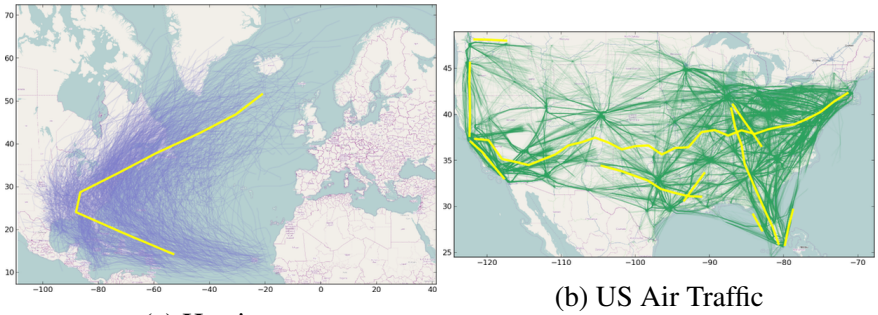


(a) 1476 Trajectories (b) 179 Trajectories (c) 181 Trajectories



(d) 210 Trajectories (e) 689 Trajectories (f) 199 Trajectories

Figure 6.8: **Atlantic Hurricanes**: (a) shows all the 1476 hurricane trajectories; (b) to (e) show the five automatically detected communities ; (b) shows hurricanes flowing from South to North. (c) and (d) show hurricanes that weaken after landfall in the US North East. (c) shows hurricanes coming from the South Atlantic Ocean and (d) shows hurricanes coming from the Mexican Gulf. (e) shows strong hurricanes that flow from the southern Atlantic Ocean to the East coast of the US, then exiting back to the Atlantic Ocean in the North. (f) shows southern Atlantic Ocean hurricanes that weakened immediately after landfall.



(a) Hurricane (b) US Air Traffic

Figure 6.9: TRACLUS Results: We show the clustering of the hurricane and air traffic datasets generated by TRACLUS [95] as a comparison. Using the suggested parameter, TRACLUS only discovers one trajectory for the hurricanes and ten trajectories for the flights.

the timings of both parameter estimation and clustering. The results are summarized in Table 6.1. TRACLUS took about one minute to estimate the parameters (MinLns $=$ $15, \varepsilon = 26$) and cluster the hurricane trajectories. Our approach took only a few seconds

	Hurricane	US Air Traffic
TRACLUS Param. Est.	36s	3600.5s (1h)
TRACLUS Clustering	35s	19993s (5.5h)
Kernel Transform	**0.2**s	**23**s
k-means Clustering		**30**s
k-NN Graph Construction	**3.6**s	
Multilevel Community Detection	**0.17**s	

Table 6.1: Performance comparison against TRACLUS [95]. Our approach performs significantly faster than TRACLUS. We are able to cluster $24K$ flight trajectories with $2M$ points in one minute, where TRACLUS took several hours.

to compute the kernel transform and detect communities among the feature vectors of the hurricane trajectories. The kernel transform and community detection were very efficient, whereas the k-nearest-neighbor graph takes the longest to compute. For the Air Traffic dataset, TRACLUS took about one hour to estimate the parameters ($\text{MinLns} = 87, \varepsilon = 20$) and took more than 5.5 hours to complete the clustering. Our approach took 23 secs to compute the kernel transform and took another 30 seconds to cluster $24K$ feature vectors.

We show results from TRACLUS in Figure 6.9. TRACLUS identifies one common sub-trajectory for the hurricanes and ten common sub-trajectories for the air-traffic dataset. TRACLUS captures the gross movement of the entire hurricane dataset, while our clusters are more refined and they show hurricanes of different shapes and at different locations. TRACLUS finds popular air routes, such as the East to West: New York to San Francisco, North to South: Chicago to Miami, and the West coast Los Angeles-San Francisco-Seattle routes. However, it does not show any clusters along the popular East coast metropolitan areas (Boston, New York, and Washington, DC). In contrast, our approach shows more refined clusters of routes and the hub airports. It shows the popular East coast routes in Figure 6.5(d).

Li *et al.* [98] have published an incremental version of the TRACLUS algorithm, however that software is not available for comparison. Their approach took seven minutes to cluster 7700 trajectories with $100K$ points on a PC with a 2.4 GHz CPU. In comparison, our approach took one minute to cluster $24K$ trajectories with $2M$ points on a similar PC.

Points	Trajectories	Total Time
4 M	168 K	195s
8 M	325 K	393s
11 M	464 K	912s

Table 6.2: Incremental Clustering of the trajectories in the San Francisco Taxi Dataset: We process the taxi dataset by incrementally loading and clustering batches of $4M$ points at a time.

6.10.2 Performance of Incrementally Clustering Taxi Trajectories

Our approach took about 15 minutes to incrementally cluster 11 million points and $464K$ trajectories. A summary of our performance is presented in Table 6.2.

6.11 Conclusions and Discussions

In this chapter, we have presented an efficient and effective approach for clustering geospatial trajectories by using the approximate kernel distance. This framework transforms trajectories into fixed-length feature vectors and it allows us to compute the approximate kernel distance by a simple dot product formulation. We have shown that this formulation is extremely efficient for clustering applications. We have combined the kernel distance and the k-means clustering methods to analyze geospatial trajectories. We have presented a two-step clustering procedure to effectively group trajectories according to their shape and spatial features. We have shown how to incrementally update the feature vectors for stream processing of evolving trajectories. In addition to k-means clustering, our approach can also be combined with community detection methods to discover unevenly distributed clusters and without the need of specifying k.

In our experiments, we show that we can cluster trajectories with millions of data points. We show results for clustering hurricane patterns, air traffic patterns, and taxi traffic patterns. Our performance is several orders of magnitude faster than the previous results [95] and shows more meaningful clusters that most of us will find interesting and appealing.

Chapter 7

Conclusions and Future Work

In this dissertation, we have shown data-driven methods for visualizing large scientific datasets. Our work has spanned the increase in dimensionality across 2D, 3D, and multi-dimensions for different applications. We have shown how to identify and visualize salient regions from very large 2D landscape images. By visually segmenting the intensity-gradient histograms, we extract meaningful structures from 3D volumetric datasets. When we visualize thousands of spatially-mapped 3D gene expression profiles, we first identify the coherently expressed genes within different brain structures. Then, we extract expression surfaces for visualizing the coherent gene expressions. We show how to combine the approximate kernel distance framework with clustering and community detection algorithms to identify common patterns among the time-varying geospatial trajectories. We believe these are the first steps towards extracting visual knowledge from huge scientific datasets with many spatial and temporal dimensions.

7.1 Future work

Very Large Images In this dissertation we have focused on large-scale landscape images that have been acquired by stitching together of a large number of photographs. However, very large scale images are finding use in a number of different areas. For example, semiconductor wafer manufacturers are using terapixel images for quality inspection of their chips for detecting circuit anomalies. As another example, latest generation microscopes stitch together volumes of very high resolution, multi-slice imagery that depicts the entire life-cycle of a number of parasites. As yet another example, astronomers are exploring the farthest reaches of the universe through the use of terapixel

imagery. Our system is currently targeted for analyzing landscape images using color and appearance. The key to analyzing other domain-specific images is to design appropriate descriptors that characterize similarity across regions of interests. The descriptors should allow fast discovery of locally distinct regions as well as accurate identification of the globally unique regions.

Volumetric Datasets We believe our hierarchical volume exploration approach naturally extends to larger datasets. The key is to perform the normalized-cut segmentation on-demand. This is critical for visualizing a large dataset because it can limit the subdivisions and segmentation to only the necessary ones. The hierarchical nature ensures that the complexity of subdividing volumes does not grow. Another possible avenue of advancement is to study other, more sophisticated measures of information content for the volumetric subdivisions. The general entropy characterizes the worst-case uncertainty. We believe this may be improved by considering the spatial structure in the volume datasets. We plan to pursue this in the future.

High-dimensional Brain Images In addition to extracting and visualizing gene expression boundaries for a single structure at a single time point, we are interested in developing visualizations for tracking the structure's physical and genetic changes in the development process. We started our pipeline with a specific brain structure; it would be interesting to construct a completely data-driven atlas by learning patterns from thousands of gene expression profiles. Another direction would be to analyze and visualize genetic information with images of different modalities, such as, MR, fMR, and EEG scan. With the development of exciting brain imaging and mapping technologies, we are interested in developing analytic and visualization tools to help improve our understanding of the most complex organ in our bodies.

Time-varying Geospatial Trajectories We are interested in developing kernel-distance-based techniques for data mining applications. We chose popular Gaussian kernel with the Euclidean distance as an example, but many other positive definite kernels can also be applied within this kernel distance framework. One possibility is to extend the Gaussian kernel with Generalized Radial Basis functions by using another distance function (like χ^2 distance). This is known to have better discriminating power in certain problem domains. We are currently working on achieving efficient feature maps for these kernels. It will be interesting to explore this space for mining rich data types such as time-series data and networks.

The main limitation of our approach lies in the high dimensionality of our feature vectors. While we have shown the effectiveness and the stream processing capabilities of the kernel distance framework, the *k*-means clustering algorithm may not be effective for high dimensional data. We have shown how to address some of the *k*-means deficiencies by using community detection algorithms, yet performing community detection in a streaming fashion is not straight forward. This is part of our future work.

7.2 Additional Contributions

In addition to the work described in this dissertation [54, 55, 56, 57], we have also contributed to other related areas during the course of this research. In the area of computational biology visualization, we have applied computational saliency to extract important timesteps from molecular simulations [82, 123]. We have summarized time-varying molecular dynamics simulations [121] with a data-driven 2D layout. We have also characterized stem cells by using their shape morphology and classified them by using support vector machines for bio-imaging applications [77]. In the area of computer graphics, we have developed a time-varying 3D social photography system [122] using smartphones. We have used the graphics-specific functions on a GPU to accelerate the generation of Poisson disk distributions [58].

List of Publications

[1] C. Y. Ip, S. Krishnan, P. Y. Yang, and A. Varshney. Fast trajectory clustering using the kernel distance. In *Intl Conference on Data Mining*, 2013. submitted.

[2] C. Y. Ip, M. Pop, and A. Varshney. Personalizing the brain atlas through gene expression correlates. In *IEEE Symposium on Biological Data Visualization*, 2013. submitted.

[3] C. Y. Ip, M. A. Yalcin, D. Luebke, and A. Varshney. Pixelpie: Maximal poisson-disk sampling with rasterization. In *High-Performance Graphics*, 2013.

[4] D. Juba, A. Cardone, C. Y. Ip, C. G. Simon, Jr, C. K. Tison, G. Kumar, M. Brady, and A. Varshney. Parallel geometric classification of stem cells by their 3d morphology. *Computational Science and Discovery*, 2013.

[5] C. Y. Ip, A. Varshney, and J. JaJa. Hierarchical exploration of volumes using multilevel segmentation of the intensity-gradient histograms. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2355–2363, 2012. doi:10.1109/TVCG.2012.231.

[6] C. Y. Ip and A. Varshney. Saliency-assisted navigation of very large landscape images. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1737–1746, 2011. doi:10.1109/TVCG.2011.231.

[7] Y. Kim, R. Patro, C. Y. Ip, D. O'Leary, A. Anishkin, S. Sukharev, and A. Varshney. Salient frame detection for molecular dynamics simulations. *Scientific Visualization: Interactions, Features, Metaphors*, 2:160–175, 2011.

[8] R. Patro, C. Y. Ip, S. Bista, S. Cho, D. Thirumalai, and A. Varshney. MDMap: A system for data-driven layout and exploration of molecular dynamics simulations. In *IEEE Symposium on Biological Data Visualization*, pages 111 –118, 2011. doi:10.1109/BioVis.2011.6094055.

[9] R. Patro, C. Y. Ip, S. Bista, and A. Varshney. Social snapshot: A system for temporally coupled social photography. *IEEE Computer Graphics & Applications*, 31(1):74 –84, 2011. doi:10.1109/MCG.2010.107.

[10] R. Patro, C. Y. Ip, and A. Varshney. Saliency guided summarization of molecular dynamics simulations. In *Scientific Visualization: Advanced Concepts*, volume 1 of *Dagstuhl Follow-Ups*, pages 321–335. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.

Bibliography

[1] H. Alt and M. Godau. Computing the fréchet distance between two polygonal curves. *Intl. Journal. of Computational Geometry & Applications*, 5:75–91, 1995.

[2] A. Andoni, P. Indyk, and R. Krauthgamer. Earth mover distance over high-dimensional spaces. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 343–352, 2008.

[3] D. Applegate, T. Dasu, S. Krishnan, and S. Urbanek. Unsupervised clustering of multidimensional distributions using earth mover distance. In *ACM SIGKDD*, pages 636–644, 2011.

[4] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. *ACM Transactions on Graphics*, 26, 2007. doi:10.1145/1276377.1276390.

[5] M. Baştan, H. Çam, U. Güdükbay, and O. Ulusoy. An MPEG-7 compatible video retrieval system with integrated support for complex multimodal queries. *IEEE Multimedia*, 17(3):62–73, 2009. doi:10.1109/MMUL.2009.74.

[6] J. A. Baerentzen and H. Aanaes. Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):243–253, 2005.

[7] C. Baillard, P. Hellier, and C. Barillot. Segmentation of brain 3D MR images using level sets and dense. *Medical Image Analysis*, 5:185–194, 2001.

[8] C. L. Bajaj, V. Pascucci, and D. R. Schikore. The contour spectrum. In *IEEE Visualization*, pages 167–173, 1997.

[9] D. Bartz, D. Mayer, J. Fischer, S. Ley, A. Rio, S. Thust, C. P. Heussel, H. U. Kauczor, and W. Straßer. Hybrid segmentation and exploration of the human lungs. In *IEEE Visualization*, page 24, 2003.

[10] M. Bello, T. Ju, J. Carson, J. Warren, W. Chiu, and I. A. Kakadiaris. Learning-based segmentation framework for tissue images containing gene expression data. *IEEE Transactions on Medical Imaging*, 26(5):728–744, 2007.

[11] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[12] U. Bordoloi and H.-W. Shen. View selection for volume rendering. In *IEEE Visualization*, pages 487 – 494, Oct 2005. doi:10.1109/VISUAL.2005.1532833.

[13] N. D. B. Bruce and J. K. Tsotsos. Saliency, attention, and visual search: An information theoretic approach. *Journal of Vision*, 9(3):1–24, 2009. doi:10.1167/9.3.5.

[14] S. Bruckner and T. Möller. Isosurface similarity maps. *Computer Graphics Forum*, 29(3):773–782, 2010. doi:10.1111/j.1467-8659.2009.01689.x.

[15] S. Bruckner and T. Möller. Isosurface similarity maps. *Computer Graphics Forum*, 29(3):773–782, 2010.

[16] M. Cabezas, A. Oliver, X. Lladó, J. Freixenet, and M. Bach Cuadra. A review of atlas-based segmentation for magnetic resonance brain images. *Computer Methods and Programs in Biomedicine*, 104(3):e158–e177, 2011.

[17] CERN. Let the number-crunching begin: the worldwide lhc computing grid celebrates first data. Press Release, 3 2008. URL: http://press.web.cern.ch/press/pressreleases/releases2008/PR13.08E.html.

[18] M. Chen and H. Jänicke. An information-theoretic framework for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1206 –1215, 2010. doi:10.1109/TVCG.2010.132.

[19] H. Childs, E. Brugger, K. Bonnell, J. Meredith, M. Miller, B. Whitlock, and N. Max. A contract based system for large data visualization. In *IEEE Visualization*, pages 191 – 198, 2005. doi:10.1109/VISUAL.2005.1532795.

[20] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *IEEE CVPR*, pages 1 –8, June 2007. doi:10.1109/CVPR.2007.383050.

[21] C. S. Co, M. A. Duchaineau, and K. I. Joy. Streaming aerial video textures. In *Scientific Visualization: Advanced Concepts*, volume 1 of *Dagstuhl Follow-Ups*, pages 336–345. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010. doi:10.4230/DFU.SciViz.2010.336.

[22] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. doi:10.1109/34.1000236.

[23] C. Correa and K.-L. Ma. Size-based transfer functions: A new volume exploration technique. *IEEE Trans. Visualization and Computer Graphics*, 14(6):1380–1387, 2008. doi:10.1109/TVCG.2008.162.

[24] C. D. Correa and K.-L. Ma. Visibility histograms and visibility-driven transfer functions. *IEEE Transactions on Visualization and Computer Graphics*, 17:192–204, 2011. doi:10.1109/TVCG.2010.35.

[25] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *Computer Vision and Pattern Recognition*, pages 1124 –1131, 2005. doi:10.1109/CVPR.2005.332.

[26] G. Daniel and M. Chen. Video visualization. In *IEEE Visualization*, pages 409–416, 2003. doi:10.1109/VISUAL.2003.1250401.

[27] M. de Hoon, S. Imoto, J. Nolan, and S. Miyano. Open source clustering software. *Bioinformatics*, 20(9):1453–1454, 2004. doi:10.1093/bioinformatics/bth078.

[28] K. Devlin, A. Chalmers, A. Wilkie, and W. Purgathofer. STAR: Tone reproduction and physically based spectral rendering. In *State of the Art Reports, Eurographics 2002*, pages 101–123, September 2002.

[29] T. Eiter and H. Mannila. Computing discrete fréchet distance. Technical report, TU Wien, 1994.

[30] K. Erleben and H. Dohlmann. *GPU Gems 3*, chapter 34 Signed Distance Fields Using Single-Pass GPU Scan Conversion of Tetrahedra. Addison-Wesley, 2004.

[31] A. C. Evans, D. L. Collins, S. Mills, E. Brown, R. Kelly, and T. M. Peters. 3D statistical neuroanatomical models from 305 MRI volumes. In *IEEE Nuclear Science Symposium and Medical Imaging Conference*, pages 1813–1817, 1993.

[32] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[33] N. Ferreira, J. T. Klosowski, C. Scheidegger, and C. Silva. Vector field k-means: Clustering trajectories by fitting multiple vector fields. In *EuroVis*, 2013.

[34] B. Fischl, D. H. Salat, E. Busa, M. Albert, M. Dieterich, C. Haselgrove, A. van der Kouwe, R. Killiany, D. Kennedy, S. Klaveness, et al. Whole brain segmentation: automated labeling of neuroanatomical structures in the human brain. *Neuron*, 33(3):341–355, 2002.

[35] T. Fogal and J. Krüger. Tuvok, an architecture for large scale volume rendering. In *International Workshop on Vision, Modeling, and Visualization*, 2010.

[36] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[37] J. Freixenet, X. Muñoz, D. Raba, J. Martí, and X. Cufí. Yet another survey on image segmentation: Region and boundary information integration. In *European Conference on Computer Vision*, pages 408–422, 2002.

[38] I. Fujishiro, T. Azuma, and Y. Takeshima. Automating transfer function design for comprehensible volume rendering based on 3d field topology analysis (case study). In *IEEE Visualization*, pages 467–470, 1999.

[39] A. R. Fuller, R. J. Zawadzki, S. Choi, D. F. Wiley, J. S. Werner, and B. Hamann. Segmentation of three-dimensional retinal image data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1719 –1726, 2007. doi:10.1109/TVCG.2007.70590.

[40] S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *ACM SIGKDD*, pages 63–72, 1999.

[41] General Electric. Flightquest. `http://www.gequest.com/c/flight`.

[42] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In *ACM SIGKDD*, pages 330–339, 2007.

[43] S. Goferman, L. Zelnik-Manor, and A. Tal. Context-aware saliency detection. In *IEEE CVPR*, pages 2376 –2383, June 2010. `doi:10.1109/CVPR.2010.5539929`.

[44] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann. Random walks for interactive organ segmentation in two and three dimensions: Implementation and validation. In *MICCAI*, pages 773–780, 2005.

[45] J. Gudmundsson, M. van Kreveld, and B. Speckmann. Efficient detection of motion patterns in spatio-temporal data sets. In *ACM Intl. workshop on Geographic information systems*, pages 250–257, 2004.

[46] D. Guo and C. Poulain. Terapixel: A spherical image of the sky. In *Microsoft Environmental Research Workshop*, 2010.

[47] A. Hanjalic, R. Lienhart, W.-Y. Ma, and J. R. Smith. The holy grail of multimedia information retrieval: So close or yet so far away? *Proceedings of the IEEE*, 96(4):541 –547, 2008. `doi:10.1109/JPROC.2008.916338`.

[48] R. A. Heckemann, J. V. Hajnal, P. Aljabar, D. Rueckert, A. Hammers, et al. Automatic anatomical brain MRI segmentation combining label propagation and decision fusion. *NeuroImage*, 33(1):115–126, 2006.

[49] J. Hernando, F. Schurmann, and L. Pastor. Towards real-time visualization of detailed neural tissue models: View frustum culling for parallel rendering. In *BioVis*, pages 25–32, 2012. `doi:10.1109/BioVis.2012.6378589`.

[50] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In *IEEE CVPR*, pages 1–8, 2007. `doi:10.1109/CVPR.2007.383267`.

[51] S. Howlett, J. Hamill, and C. O'Sullivan. Predicting and evaluating saliency for simplified polygonal models. *ACM Trans. Applied Perception*, 2:286–308, 2005. `doi:10.1145/1077399.1077406`.

[52] R. Huang and K.-L. Ma. Rgvis: Region growing based techniques for volume visualization. In *Pacific Graphics*, pages 355–363, 2003.

[53] M. Hussein, A. Varshney, and L. Davis. On implementing graph cuts on cuda. In *General Purpose Processing on Graphics Processing Units*, 2007.

[54] C. Y. Ip, S. Krishnan, P. Y. Yang, and A. Varshney. Fast trajectory clustering using the kernel distance. In *Intl Conference on Data Mining*, 2013. submitted.

[55] C. Y. Ip, M. Pop, and A. Varshney. Personalizing the brain atlas through gene expression correlates. In *IEEE Symposium on Biological Data Visualization*, 2013. submitted.

[56] C. Y. Ip and A. Varshney. Saliency-assisted navigation of very large landscape images. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1737–1746, 2011. doi:10.1109/TVCG.2011.231.

[57] C. Y. Ip, A. Varshney, and J. JaJa. Hierarchical exploration of volumes using multilevel segmentation of the intensity-gradient histograms. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2355–2363, 2012. doi:10.1109/TVCG.2012.231.

[58] C. Y. Ip, M. A. Yalcin, D. Luebke, and A. Varshney. Pixelpie: Maximal poisson-disk sampling with rasterization. In *High-Performance Graphics*, 2013.

[59] L. Itti and C. Koch. A comparison of feature combination strategies for saliency-based visual attention systems. *Journal of Electronic Imaging*, 10:161–169, 2001. doi:10.1117/1.1333677.

[60] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998. doi:10.1109/34.730558.

[61] Z. Ivezic et al. LSST: from Science Drivers to Reference Design and Anticipated Data Products. *ArXiv e-prints*, 2008.

[62] D. W. Jacobs. What makes viewpoint-invariant properties perceptually salient? *Journal of the Optical Society of America A*, 20(7):1304–1320, 2003.

[63] H. Jänicke, M. Bottinger, U. Mikolajewicz, and G. Scheuermann. Visual exploration of climate variability changes using wavelet analysis. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1375–1382, 2009.

[64] H. Jänicke and M. Chen. A salience-based quality metric for visualization. *Computer Graphics Forum*, 29(3):1183–1192, 2010. doi:10.1111/j.1467-8659.2009.01667.x.

[65] H. Jänicke and G. Scheuermann. Measuring complexity in lagrangian and eulerian flow descriptions. *Computer Graphics Forum*, 29(6):1783–1794, 2010. doi:10.1111/j.1467-8659.2010.01648.x.

[66] H. Jänicke and G. Scheuermann. Visual analysis of flow features using information theory. *IEEE Computer Graphics & Applications*, 30:40–49, 2010. doi:10.1109/MCG.2010.17.

[67] H. Jänicke, A. Wiebel, G. Scheuermann, and W. Kollmann. Multifield visualization using local statistical complexity. *IEEE Trans. Visualization and Computer Graphics*, 13(6):1384 –1391, 2007. doi:10.1109/TVCG.2007.70615.

[68] F. Janoos, M. O. Irfanoglu, K. Mosaliganti, R. Machiraju, K. Huang, P. Wenzel, A. DeBruin, and G. Leone. Multi-resolution image segmentation using the 2-point correlation functions. In *ISBI Biomedical Imaging*, pages 300–303, 2007.

[69] F. Janoos, K. Mosaliganti, X. Xu, R. Machiraju, K. Huang, and S. T. Wong. Robust 3D reconstruction and identification of dendritic spines from optical microscopy imaging. *Medical image analysis*, 13(1):167, 2009.

[70] F. Janoos, B. Nouansengsy, X. Xu, R. Machiraju, and S. T. Wong. Classification and uncertainty visualization of dendritic spines from optical microscopy imaging. *Computer Graphics Forum*, 27(3):879–886, 2008.

[71] C. S. Jensen, D. Lin, and B. C. Ooi. Continuous clustering of moving objects. *IEEE Transactions on Knowledge and Data Engineering*, 19(9):1161–1174, 2007.

[72] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment*, 1(1):1068–1080, 2008.

[73] C. Johnson. Top scientific visualization research problems. *IEEE Computer Graphics & Applications*, 24(4):13 – 17, 2004. doi:10.1109/MCG.2004.20.

[74] A. Jones, C. Overly, and S. Sunkin. The allen brain atlas: 5 years and beyond. *Nature Reviews Neuroscience*, 10(11):821–828, 2009.

[75] S. Joshi, R. V. Kommaraji, J. M. Phillips, and S. Venkatasubramanian. Comparing distributions and shapes using the kernel distance. In *ACM Symposium on Computational Geometry*, pages 47–56, 2011. doi:10.1145/1998196.1998204.

[76] T. Ju, J. Warren, G. Eichele, C. Thaller, W. Chiu, and J. Carson. A geometric database for gene expression data. In *Symposium on Geometry Processing*, pages 166–176, 2003.

[77] D. Juba, A. Cardone, C. Y. Ip, C. G. Simon, Jr, C. K. Tison, G. Kumar, M. Brady, and A. Varshney. Parallel geometric classification of stem cells by their 3d morphology. *Computational Science and Discovery*, 2013.

[78] I. A. Kakadiaris, M. Bello, S. Arunachalam, W. Kang, T. Ju, J. Warren, J. Carson, W. Chiu, C. Thaller, and G. Eichele. Landmark-driven, atlas-based segmentation of mouse brain tissue images containing gene expression data. In *MICCAI*, pages 192–199, 2004.

[79] M. Kazhdan and H. Hoppe. Streaming multigrid for gradient-domain operations on large images. *ACM Transactions on Graphics*, 27:1–10, 2008. doi:10.1145/1360612.1360620.

[80] G. Kim and A. Torralba. Unsupervised Detection of Regions of Interest using Iterative Link Analysis. In *NIPS*, 2009.

[81] J. Kim and J. JaJa. Information-aware 2^n-tree for efficient out-of-core indexing of very large multidimensional volumetric data. In *International Conference on Scientific and Statistical Database Management*, July 2007. doi:10.1109/SSDBM.2007.15.

[82] Y. Kim, R. Patro, C. Y. Ip, D. O'Leary, A. Anishkin, S. Sukharev, and A. Varshney. Salient frame detection for molecular dynamics simulations. *Scientific Visualization: Interactions, Features, Metaphors*, 2:160–175, 2011.

[83] Y. Kim and A. Varshney. Saliency-guided enhancement for volume visualization. *IEEE Trans. Visualization and Computer Graphics*, 12(5):925 –932, 2006. doi:10.1109/TVCG.2006.174.

[84] Y. Kim and A. Varshney. Persuading visual attention through geometry. *IEEE Transactions on Visualization and Computer Graphics*, 14:772–782, July 2008. doi:10.1109/TVCG.2007.70624.

[85] Y. Kim, A. Varshney, D. W. Jacobs, and F. Guimbretière. Mesh saliency and human eye fixations. *ACM Trans. Applied Perception*, 7:12:1–12:13, 2010. doi:10.1145/1670671.1670676.

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

[87] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8:270–285, 2002. doi:10.1109/TVCG.2002.1021579.

[88] J. Kniss and G. Wang. Supervised manifold distance segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1637 –1649, 2011. doi:10.1109/TVCG.2010.120.

[89] I. S. Kohane, J. M. Drazen, and E. W. Campion. A glimpse of the next 100 years in medicine. *The New England journal of medicine*, 367(26):2538, 2012.

[90] J. Kopf, M. Uyttendaele, O. Deussen, and M. Cohen. Capturing and viewing gigapixel images. *ACM Trans. Graphics*, 26(3), 2007. doi:10.1145/1276377.1276494.

[91] D. Kotz and T. Henderson. CRAWDAD: A community resource for archiving wireless data at dartmouth. *IEEE Pervasive Computing*, 4(4):12–14, 2005. doi:10.1109/MPRV.2005.75.

[92] P.-Y. Laffont, J. Y. Jun, C. Wolf, Y.-W. Tai, K. Idrissi, G. Drettakis, and S.-e. Yoon. Interactive content-aware zooming. In *Proceedings of Graphics Interface 2010*, pages 79–87, 2010.

[93] C. H. Lee, Y. Kim, and A. Varshney. Saliency-guided lighting. *IEICE Trans. Information and Systems*, E92.D(2):369–373, 2009. doi:10.1587/transinf.E92.D.369.

[94] C. H. Lee, A. Varshney, and D. W. Jacobs. Mesh saliency. *ACM Trans. Graphics*, 24:659–666, July 2005. doi:10.1145/1073204.1073244.

[95] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *ACM SIGMOD*, pages 593–604, 2007.

[96] E. S. Lein, M. J. Hawrylycz, N. Ao, M. Ayres, A. Bensinger, A. Bernard, A. F. Boe, M. S. Boguski, K. S. Brockway, E. J. Byrnes, et al. Genome-wide atlas of gene expression in the adult mouse brain. *Nature*, 445(7124):168–176, 2006.

[97] Z. Li, B. Ding, J. Han, and R. Kays. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3(1-2):723–734, 2010.

[98] Z. Li, J.-G. Lee, X. Li, and J. Han. Incremental clustering for trajectories. In *Database Systems for Advanced Applications*, pages 32–46, 2010.

[99] H. Ling and K. Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):840–853, 2007.

[100] T. Liu, J. Sun, N.-N. Zheng, X. Tang, and H.-Y. Shum. Learning to detect a salient object. In *IEEE CVPR*, pages 1 –8, June 2007. doi:10.1109/CVPR.2007.383047.

[101] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 163–169, July 1987.

[102] Q. Luan, S. M. Drucker, J. Kopf, Y.-Q. Xu, and M. F. Cohen. Annotating gigapixel images. In *UIST*, pages 33–36, 2008. doi:10.1145/1449715.1449722.

[103] S. Ma, Y. Zheng, and O. Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *ICDE*, pages 140–149, 2013.

[104] R. Machiraju, J. E. Fowler, D. Thompson, and a. B. S. W. Schroeder. Evita: Efficient visualization and interrogation of tera-scale data. In *Data mining for scientific and engineering applications*. Kluwer, 2001.

[105] R. Maciejewski, I. Woo, W. Chen, and D. Ebert. Structuring feature space: A non-parametric method for volumetric transfer function generation. *IEEE Transactions on Visualization and Computer Graphics*, 15:1473–1480, 2009. doi:10.1109/TVCG.2009.185.

[106] E. Mardis. A decade/'s perspective on dna sequencing technology. *Nature*, 470(7333):198–203, 2011.

[107] R. McLendon et al. Comprehensive genomic characterization defines human glioblastoma genes and core pathways. *Nature*, 455(7216):1061–1068, 2008.

[108] J. Meyer-Spradow, T. Ropinski, J. Mensmann, and K. Hinrichs. Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations. *IEEE Computer Graphics and Applications*, 29(6):6–13, 2009. doi:10.1109/MCG.2009.130.

[109] G. S. Miller. The definition and rendering of terrain maps. *ACM SIGGRAPH Computer Graphics*, 20(4):39–48, 1986.

[110] K. Mosaliganti, F. Janoos, O. Irfanoglu, R. Ridgway, R. Machiraju, K. Huang, J. Saltz, G. Leone, and M. Ostrowski. Tensor classification of *N* point correlation function features for histology tissue segmentation. *Medical image analysis*, 13(1):156–166, 2009.

[111] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP*, pages 331–340, 2009.

[112] M. Nanni and D. Pedreschi. Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3):267–289, 2006.

[113] M. E. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[114] NOAA. National hurricane center data archive. http://www.nhc.noaa.gov/pastall.shtml.

[115] S. D. Olabarriaga and A. W. M. Smeulders. Interaction in the segmentation of medical images: A survey. *Medical image analysis*, 5(2):127–142, 2001.

[116] A. Oliva, A. Torralba, M. Castelhano, and J. Henderson. Top-down control of visual attention in object detection. In *IEEE ICIP*, pages 253–6, Sept 2003. doi:10.1109/ICIP.2003.1246946.

[117] J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, and J. Phillips. GPU computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

[118] J. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. Lefohn, and T. Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.

[119] D. Page, Y. Sun, A. Koschan, J. Paik, and M. Abidi. Normal vector voting: crease detection and curvature estimation on large, noisy meshes. *Graphical Models*, 64(3):199–229, 2002.

[120] D. P. Panda and A. Rosenfeld. Image segmentation by pixel classification in (gray level, edge value) space. *IEEE Transactions on Computers*, 100(9):875–879, 1978.

[121] R. Patro, C. Y. Ip, S. Bista, S. Cho, D. Thirumalai, and A. Varshney. Mdmap: A system for data-driven layout and exploration of molecular dynamics simulations. In *IEEE Symposium on Biological Data Visualization*, pages 111 –118, 2011. doi:10.1109/BioVis.2011.6094055.

[122] R. Patro, C. Y. Ip, S. Bista, and A. Varshney. Social snapshot: A system for temporally coupled social photography. *IEEE Computer Graphics & Applications*, 31(1):74 –84, 2011. doi:10.1109/MCG.2010.107.

[123] R. Patro, C. Y. Ip, and A. Varshney. Saliency guided summarization of molecular dynamics simulations. In *Scientific Visualization: Advanced Concepts*, volume 1 of *Dagstuhl Follow-Ups*, pages 321–335. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.

[124] N. Pelekis, I. Kopanakis, E. E. Kotsifakos, E. Frentzos, and Y. Theodoridis. Clustering uncertain trajectories. *Knowledge and information systems*, 28(1):117–147, 2011.

[125] R. Perley, C. Chandler, B. Butler, and J. Wrobel. The expanded very large array: A new telescope for new science. *The Astrophysical Journal Letters*, 739:L1, 2011.

[126] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. S. Avila, K. Martin, R. Machiraju, and J. Lee. The transfer function bake-off. *IEEE Computer Graphics & Applications*, 21:16–22, 2001. doi:10.1109/38.920623.

[127] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. A parsimonious model of mobile partitioned networks with clustering. In *Communication Systems and Networks and Workshops COMSNETS*, pages 1 –10, 2009. doi:10.1109/COMSNETS.2009.4808865.

[128] J.-S. Prassni, T. Ropinski, and K. Hinrichs. Uncertainty-aware guided volume segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1358 –1365, 2010. doi:10.1109/TVCG.2010.208.

[129] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan kaufmann, 1993.

[130] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20:1177–1184, 2007.

[131] S. Roettger, M. Bauer, and M. Stamminger. Spatialized transfer functions. In *EuroVis*, pages 271–278, 2005.

[132] T. Rohlfing, R. Brandt, R. Menzel, C. R. Maurer Jr, et al. Evaluation of atlas selection strategies for atlas-based image segmentation with application to confocal microscopy images of bee brains. *NeuroImage*, 21(4):1428–1442, 2004.

[133] M. Rubinstein, D. Gutierrez, O. Sorkine, and A. Shamir. A comparative study of image retargeting. *ACM Transactions on Graphics*, 29(6), 2010. doi:10.1145/1882261.1866186.

[134] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *Intl. Journal of Computer Vision*, 40(2):99–121, 2000.

[135] M. Ruiz, A. Bardera, I. Boada, I. Viola, M. Feixas, and M. Sbert. Automatic transfer functions based on informational divergence. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1932–1941, 2011. doi:10.1109/TVCG.2011.173.

[136] B. Russell, W. Freeman, A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *IEEE CVPR*, pages 1605 – 1614, 2006. doi:10.1109/CVPR.2006.326.

[137] C. R. Salama, M. Keller, and P. Kohlmann. High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1021–1028, 2006.

[138] J. Sankaranarayanan, H. Samet, and A. Varshney. A fast all nearest neighbor algorithm for applications involving large point-clouds. *Computers & Graphics*, 31(2):157 – 174, 2007. doi:10.1016/j.cag.2006.11.011.

[139] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[140] M. A. Selver and C. Güzeliç. Semiautomatic transfer function initialization for abdominal visualization using self-generating hierarchical radial basis function networks. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):395–409, 2009.

[141] P. Sereda, A. Bartroli, I. Serlie, and F. Gerritsen. Visualization of boundaries in volumetric data sets using LH histograms. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):208–218, 2006.

[142] E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt. Hierarchy and adaptivity in segmenting visual scenes. *NATURE*, 442(7104):810, 2006.

[143] A. Sherbondy, M. Houston, and S. Napel. Fast volume segmentation with simultaneous visualization using programmable graphics hardware. In *IEEE Visualization*, page 23, 2003.

[144] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[145] S. Shirdhonkar and D. W. Jacobs. Approximate earth mover's distance in linear time. In *IEEE Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[146] C. Sigg, R. Peikert, and M. Gross. Signed distance transform using graphics hardware. In *IEEE Visualization*, pages 83–90, 2003.

[147] A. Sud, M. A. Otaduy, and D. Manocha. Difi: Fast 3D distance field computation using graphics hardware. *Computer Graphics Forum*, 23(3):557–566, 2004.

[148] P. Sudmant, J. Kitzman, F. Antonacci, C. Alkan, M. Malig, A. Tsalenko, N. Sampas, L. Bruhn, J. Shendure, E. Eichler, et al. Diversity of human copy number variation and multicopy genes. *Science*, 330(6004):641–646, 2010.

[149] B. Suh, H. Ling, B. B. Bederson, and D. W. Jacobs. Automatic thumbnail cropping and its effectiveness. *CHI Letters (UIST 2003)*, 5(2):95–104, 2003.

[150] B. Summa, G. Scorzelli, M. Jiang, P.-T. Bremer, and V. Pascucci. Interactive editing of massive imagery made simple: Turning Atlanta into Atlantis. *ACM Transactions on Graphics*, 30:1–13, 2011. doi:10.1145/1944846.1944847.

[151] S. M. Sunkin and J. G. Hohmann. Insights from spatially mapped gene expression in the mouse brain. *Human molecular genetics*, 16(R2):R209–R219, 2007.

[152] S. M. Sunkin, L. Ng, C. Lau, T. Dolbeare, T. L. Gilbert, C. L. Thompson, M. Hawrylycz, and C. Dang. Allen brain atlas: an integrated spatio-temporal portal for exploring the central nervous system. *Nucleic acids research*, 41(D1):D996–D1008, 2013.

[153] A. S. Szalay. The sloan digital sky survey. *Computing in Science and Engineering*, 1(2):54–62, 1999. doi:10.1109/5992.753047.

[154] C.-K. Tang and G. Medioni. Extremal feature extraction from 3-d vector and noisy scalar fields. In *IEEE Visualization*, pages 95–102, 1998.

[155] C.-K. Tang and G. Medioni. Robust estimation of curvature information from noisy 3D data for shape description. In *ICCV*, volume 1, pages 426–433, 1999.

[156] L.-A. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, C.-C. Hung, and W.-C. Peng. On discovery of traveling companions from streaming trajectories. In *ICDE*, pages 186–197, 2012. doi:10.1109/ICDE.2012.33.

[157] C. C. Tanner, C. J. Migdal, and M. T. Jones. The clipmap: a virtual mipmap. In *SIGGRAPH*, pages 151–158, 1998. doi:10.1145/280814.280855.

[158] T. Torsney-Weir, A. Saad, T. Moller, H. Hege, B. Weber, J. Verbavatz, and S. Bergner. Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1892 –1901, 2011. doi:10.1109/TVCG.2011.248.

[159] J. K. Tsotsos, S. M. Culhane, W. Y. K. Wai, Y. Lai, N. Davis, and F. Nuflo. Modeling visual-attention via selective tuning. *Artificial Intelligence*, 78(1-2):507–545, 1995. doi:10.1016/0004-3702(95)00025-9.

[160] F.-Y. Tzeng, E. B. Lum, and K.-L. Ma. An intelligent system approach to higher-dimensional classification of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 11:273–284, 2005. doi:10.1109/TVCG.2005.38.

[161] F.-Y. Tzeng and K.-L. Ma. A cluster-space visual interface for arbitrary dimensional classification of volume data. In *VisSym*, pages 17–24, 2004.

[162] M. Vaillant and J. Glaunes. Surface matching via currents. In *Information Processing in Medical Imaging*, pages 1–5, 2005.

[163] I. Viola, M. Feixas, M. Sbert, and M. Gröller. Importance-driven focus of attention. *IEEE Trans. Visualization and Computer Graphics*, 12(5):933 –940, 2006. doi:10.1109/TVCG.2006.152.

[164] Y. Wan, H. Otsuna, C.-B. Chien, and C. Hansen. Interactive extraction of neural structures with user-guided morphological diffusion. In *BioVis*, pages 1–8, 2012. doi:10.1109/BioVis.2012.6378577.

[165] W. Wang, Y. Wang, Q. Huang, and W. Gao. Measuring visual saliency by site entropy rate. In *IEEE CVPR*, pages 2368 –2375, June 2010. doi:10.1109/CVPR.2010.5539927.

[166] Y. Wang, W. Chen, G. Shan, T. Dong, , and X. Chi. Volume exploration using ellipsoidal gaussian transfer functions. In *PacificVis*, pages 25–32, 2010. doi:10.1109/PACIFICVIS.2010.5429612.

[167] Y. Wang, W. Chen, J. Zhang, T. Dong, G. Shan, and X. Chi. Efficient volume exploration using the gaussian mixture model. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1560 –1573, 2011. doi:10.1109/TVCG.2011.97.

[168] A. Wiebel, X. Tricoche, D. Schneider, H. Jänicke, and G. Scheuermann. Generalized streak lines: Analysis and visualization of boundary induced vortices. *IEEE Trans. Visualization and Computer Graphics*, 13(6):1735 –1742, 2007. doi:10.1109/TVCG.2007.70557.

[169] M. Wikelski and R. Kays. Movebank: archive, analysis and sharing of animal movement data. http://www.movebank.org.

[170] R. Wolz, P. Aljabar, J. V. Hajnal, A. Hammers, and D. Rueckert. Leap: Learning embeddings for atlas propagation. *NeuroImage*, 49(2):1316–1325, 2010.

[171] S. X. Yu and J. Shi. Multiclass spectral clustering. In *International Conference on Computer Vision*, pages 313–319, 2003. doi:10.1109/ICCV.2003.1238361.

[172] J. Yuan, Y. Zheng, X. Xie, and G. Sun. T-Drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):220–232, 2013.

[173] Q. Zhang and X. Lin. Clustering moving objects for spatio-temporal selectivity estimation. In *Australasian database conference*, pages 123–130, 2004.

[174] K. Zheng, Y. Zheng, N. J. Yuan, and S. Shang. On discovery of gathering patterns from trajectories. In *ICDE*, 2013.

[175] Y. Zheng, X. Xie, and W.-Y. Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Engineering Bulletin*, 33(2):32–40, 2010.

[176] Y. Zheng and X. Zhou, editors. *Computing with Spatial Trajectories*. Springer, 2011.

[177] J. Zhou and M. Takatsuka. Automatic transfer function generation using contour tree controlled residue flow model and color harmonics. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1481 –1488, 2009. doi:10.1109/TVCG.2009.120.

[178] L. Zhukov, K. Museth, D. Breen, A. H. Barr, and R. Whitaker. Level set modeling and segmentation of diffusion tensor magnetic resonance imaging brain data. *Journal of Electronic Imaging*, 12(1):125–133, 2003.

[179] R. Zonenschein and L. Velho. Visorama 2.0: a platform for multimedia gigapixel panoramas. In *SIGGRAPH ASIA 2010 Posters*, 2010. doi:10.1145/1900354.1900424.