SCALABLE, SITUATIONALLY AWARE VISUAL ANALYTICS AND APPLICATIONS

by

Todd Eaglin

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2017

Approved by:

Dr. William Ribarsky

Dr. Xiaoyu Wang

Dr. Zach Wartell

Dr. Eric Delmelle

ABSTRACT

TODD EAGLIN. Scalable, situationally aware visual analytics and applications.
(Under the direction of DR. WILLIAM RIBARSKY)

There is a need to understand large and complex datasets to provide better situational awareness in-order to make timely well-informed actionable decisions in critical environments. These types of environments include emergency evacuations for large buildings, indoor routing for buildings in emergency situations, large-scale critical infrastructure for disaster planning and first responders, LiDAR analysis for coastal planning in disaster situations, and social media data for health related analysis. I introduce novel work and applications in real-time interactive visual analytics in these domains. I also detail techniques, systems and tools across a range of disciplines from GPU computing for real-time analysis to machine learning for interactive analysis on mobile and web-based platforms.

ACKNOWLEDGMENTS

"If you're going to try, go all the way. Otherwise, don't even start. This could mean losing girlfriends, wives, relatives and maybe even your mind. It could mean not eating for three or four days. It could mean freezing on a park bench. It could mean jail. It could mean derision. It could mean mockery–isolation. Isolation is the gift. All the others are a test of your endurance, of how much you really want to do it. And, you'll do it, despite rejection and the worst odds. And it will be better than anything else you can imagine. If you're going to try, go all the way. There is no other feeling like that. You will be alone with the gods, and the nights will flame with fire. You will ride life straight to perfect laughter. It's the only good fight there is."

— Charles Bukowski, Factotum

Towards the end of my dissertation this quote by Charles Bukowski stuck with me very poignantly. Completing a PhD is no easy feat. It is wrought with struggle, isolation, self doubt and an entire list of other emotions. It really takes a piece of who you are. No one will really understand unless they have gone through the gauntlet themselves, but you have to endure. You have to keep going if you want to finish. I endured many difficult struggles, from changing advisors half way into my dissertation to my new advisor passing away right near the end. Through it all you have to keep going.

There are many people I have to thank who helped me along this journey. To my parents and family who were always there. To my brother Adam who proof read parts of my dissertation. To my close friends; Dan Lemmon, whom I have known for well over a decade. You saw me at my worst and always gave me a pep talk when I needed it most. Spencer Kilgore, you always kept motivating me to give it my all even when I was struggling, tired and didn't want to keep going. To my committee for helping me finish my dissertation after Dr. Ribarsky's passing. It was a struggle for me and I appreciated all the help.

But, most of all to my late advisor Dr. William Ribarsky, he unfortunately passed away three and a half weeks right before my dissertation defense. Dr. Ribarsky helped me so much along the way. After switching advisors about two years into my program I had contemplating quitting, but having him as my advisor gave me confidence to keep going. This work was as much his as it was mine and he will be sorely missed. Thank you for everything you did for me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

The challenge of Big Data is multifaceted with the end goal of extracting actionable knowledge. Big data is a powerful tool in today's world, because decision makers can use it to garner knowledge, make critical decisions and optimize existing processes. But unlocking the potential of big data first requires us to solve a number of challenges. It requires the integration of techniques, technologies, and visualizations to uncover hidden meaning from datasets that are diverse, complex, and of a massive scale. Big data is beyond the ability of commodity software tools to manage, process, and visualize in a reasonable amount of time. For example, Wal-Mart handles more than one million customer transactions every hour. Facebook handles 40 billion photos from its user base. Therefore, new techniques, technologies, and applications must be developed in order to provide scalable, situational awareness for decision makers [18, 66].

The Big Data challenge can be broken down into several aspects, but for the focus of our work, I emphasize the following:

- *Volume* - The sheer scale of data has become a significant challenge. With the advent of mobile devices, the internet of things and connected sensors data is growing at an exponential rate.

- *Velocity* - This is defined as the the speed at which data is being collected from sensors and smart metering. The challenge here is being able to manage and react quickly to new data in real time.

- *Variety* - Data comes in numerous unstructured and structured formats like video, images, text documents, audio and spatial. Managing all these different types of formats is a difficult problem.
- *Value* - The last part of Big data is identifying value from the data.

The purpose of this dissertation is to contribute novel solutions for solving some of these problems in specific domains. Since these problems are multifaceted our novel solutions tackle data, analytics, scale, visualization and interaction to provide insight, and actionable knowledge. I focus on four main application domains whereby I have made significant contributions.

- *Emergency Response and Evacuation Simulations* - Evacuating large buildings or stadiums is an important need for planners, but the current solutions are impractical for real-world situations, or handling scenarios with changing elements. To support this work tools and systems had to be developed. Complex building models and graph techniques had to be developed to create a real-time application. Multiple visualizations and reporting tools had to be developed in-order to provide accurate and important information to planners. Interactions must be developed to allow new situations to be evaluated and easily explored. We refer to these applications later as *EvacSim* for the evacuation simulations and Effective Emergency Response Communication *EERC*
- *Hurricane Simulations on Critical Infrastructure* - Nowhere is the need to understand large heterogeneous datasets more important than in disaster monitoring and emergency response, where critical decisions have to be made in a timely fashion and the discovery of important events requires an understanding of a collection of complex

simulations. To gain enough insights for actionable knowledge, the development of models and analysis of modeling results usually requires that models be run many times so that all possibilities can be covered. Central to the goal of our research is, therefore, the use of ensemble visualization of a large scale simulation space to appropriately aid decision makers in reasoning about infrastructure behaviors and vulnerabilities in support of critical infrastructure analysis. We refer to this application later as $HurricaneVA$.

- *Remote Sensing* - LiDAR data is an important data source for understanding geospatial features in urban planning, land use analysis, emergency response, and other applications. But it presents several complex challenges that require novel solutions. The first is scale, LiDAR data is massive and hard to manage. Second, it's hard to interact with through visual analytics. Lastly, there is no searchable content. We refer to this application later as $LiDARVA$.

- *Social Media* - A major health problem gripping universities in the US is binge drinking. College students often post drinking related texts and images on social media as a socially desirable identity. But, this behavior can lead to poor school performance, health related issues, and even death. Understanding how to identify these patterns can lead to tools to assist students to make healthier choices.

This dissertation is organized as follows. In Chapter 2 different types of data and analytical techniques are discussed with respect to the four mentioned domains. In Chapter 3 I discuss the visualizations utilized and developed. In Chapter 4, I present novel solutions to tackle scaling data analytics and visualization for these specific problems. In Chapter 5 I detail the us of mobile and web-based platforms and interaction unique to those platforms.

Lastly in Chapter 6 I conclude with user study and case study results.

CHAPTER 2: DATA AND ANALYTICS

The challenge of big data is a difficult problem. There exists lots of different types of data that can be unstructured, heterogenous, and multi-variate in nature [135]. This poses a significant challenge because there are so many different types and each type might entail a unique problem that needs to be solved. In order to solve these problems patterns and meaning have to be extracted before actionable knowledge can be made. Therefore, different strategies have to be developed. In this chapter I focus on our contributions to data and analytics through specific domains. These areas include spatio-temporal, graphs and networks, remote sensing, crowd source, social media, and simulations.

2.1 SPATIAL AND TEMPORAL (STKDE)

The first major type of data I focus on is spatio-temporal, although it can either be separate or combined, I focus on the latter. Temporal, geospatial, and spatio-temporal data are the most prominent and ubiquitous data types in visual analytics [72]. Analytics plays an important role in the understanding of geospatial data [3]. There is also need to find clusters and hotspots in space-time in order to drive further analysis [31]. For these reasons finding patterns and areas of interest in spatio-temporal data is necessary for many problem domains [73]. But, it can be extremely difficult and computationally challenging to analyze, visualize and extract meaning from spatio-temporal data in a real-time application. Scalability and complexity of the data present new challenges for effective analysis. Spatio-temporal data also requires new interactions, visualization and computational algorithims all of which I discuss in later chapters.

For the puposes of this dissertation one technique in particular that I focus on is the Kernel Density Estimation (KDE). Using a KDE is an effective way to analyze geospatial hotspots [74]. Although additional techniques exist for analyzing temporal and spatial data our original contributions to this field focus on the KDE technique. These density estimations can be applied to spatio-temporal data in order to find clusters and behavior across space and time, but there are several critical limitations. Delmelle et al. [30] explored the use of the space-time kernel density estimation (STKDE) on dengue fever outbreak in Cali, Columbia. This work demonstrated the computation challenge, but also the potential payoff, of performing the STKDE.

$$f\left(x,y,t\right)=\frac{1}{nh_s^2h_t}\sum_i I\left(d_i<h_s,t_i<h_t\right)k_s\left(\frac{x-x_i}{h_s},\frac{y-y_i}{h_s}\right)k_t\left(\frac{t-t_i}{h_t}\right) \tag{1}$$

The STKDE works by discretizing a 3D space-time cube into voxels. For each voxel in a 3D region the density is calculated based on surrounding data points that fall within a spatial and temporal bandwidth, which are then weighted using an Epanechnikov kernel function as indicated in Equation 1. For each voxel given the coordinates x, y, z the density is estimated based on the surrounding data points given as x_i, y_i, z_i. The spatial and temporal distances between the voxel and data point is given by d_i and t_i. The spatial and temporal bandwidths are given by h_s and h_t.

2.2 GRAPHS AND NETWORKS

Another important type of data is graphs or networks. Graph analytics can be used to search, discover or make recommendations. Graphs are quite common and most graph analysis techniques have been well established, but the challenge is the scale of graphs. Analyzing massive graphs from social media like social networks is a big data challenge.

I utilize graph networks significantly in the later sections, but detail some background information here. In the later sections I utilize networks for remote sensing, hurricane simulations, evacuation simulations and emergency response.

Schwartz et al. [125] looked at discovering shared interests through graph analytics. Gupta et al. [55] detailed the process by which Twitter's WTF (Who to Follow?) recommendation service works. It is responsible for generating millions of connections daily between users with common connections and shared interests. They examined several analysis approaches Singular Value Decomposition, Random Walks and Cosine Similarity. They also chose to store the entire Twitter user graph in memory. To do so requires significant computational resources. John et al. [65] investigated graph analysis on the brain connectivity of patients with early Alzheimer's disease.

Liu et al [89] developed a visual analytics application to generate graph networks for multi-floor indoor buildings. Using these networks they were then able to perform indoor routing and emergency response. Guest et al. [54] took their work further and used the building graphs to run large scale building evacuation simulations using shortest path graph algorithms. They later constructed campus wide graphs, but had scaling issues due to the high computational complexity.

2.3 REMOTE SENSING (LiDARVA)

For the purpose of this work I only focus on one particular type of remote sensing data. I acknowledge there are other types, but overall the strategies remain consistent. Therefore, I only discuss work done on LiDAR data. LiDAR, a remote sensing technology, has become a crucial instrument for us to extract and monitor up-to-date, accurate geographic information about areas of interests [138]. Professionals involved in a variety of industries and applications (like disaster management) are increasingly utilizing three-dimensional

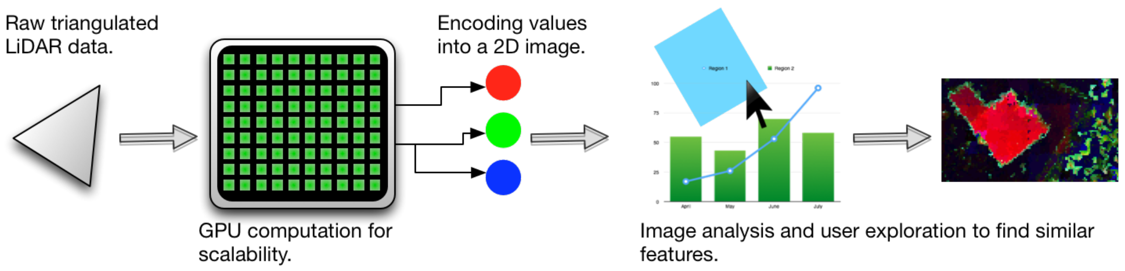

Raw triangulated LiDAR data.
GPU computation for scalability.
Encoding values into a 2D image.
Image analysis and user exploration to find similar features.

Figure 1: This figure shows the overall architecture of our interactive visual analytics systems for understanding LiDAR data.

sources of information to create photorealistic 3D visualizations, extract 3D features and export products to geospatial tools to help understand geospatial features in both urban planning [15] and emergency responding [43].

Especially in emergency management and disaster response, LiDAR presents a more effective data source to locate weak points. For instance, Kwan et al. [83] used about 50 million LiDAR points to identify blockages in the transportation network caused by Hurricane Katrina. Whereas, Clasen et al. [24] further examined the potential areas where LiDAR data can be used for disaster and emergency planning, including understanding how slope of the terrain affects landslides, analyzing tree maps for fire disasters, and more importantly finding suitable areas for rescuer deployment and helicopter landings.

Butkiewicz et al. [11] explored the combination of visual analytics and LiDAR to detect temporal changes in a city environment. The authors looked at finding the differences between different time periods of LiDAR data using the 3D geometry created from a LiDAR point cloud. Using the computed geometry provided more accuracy than a point cloud using a grid approach with nearest neighbor and outputting that as a 2D image.

Butkiewicz et al. [11] did not mention anything about scalability since they compared unique spatial regions in time and did not compare any deltas between regions. This resulted in a much smaller number of overall comparisons. The authors concluded that there is still much work to be done in the area of algorithms for handling issues of matching or issues with users not being able to define filter metrics. Blaschke et al. [6] surveyed the existing research in image analysis for remote sensing; they outlined some of the benefits to using image segmentation. A lot of work has been done on classification in LiDAR data and extraction of specific features like buildings, as described by Hermosilla et al. [58], who utilized image thresholding using two specific values. The first threshold value is the minimum height of a particular building and the second value indicates the presence of vegetation. Their results were quite strong, using image thresh- olding, but this was only when the thresholding values were adjusted to the specific type of environment in the LiDAR data.

While the utility of LiDAR can be significant, analyzing it effectively over large areas still comes with significant challenges. Based on our collaboration with the North Carolina Department of Transportation, I observed the following challenges:

- *Scale* - Due to the nature of LiDAR it can grow exponentially when examining larger spatial regions, at higher resolutions, and at different points in time. The files can be hard to manage; even loading and viewing them can be a problem. Collecting different sources of data, processing them with a myriad of different tools, and converting LiDAR coordinate systems can also be quite time-consuming. Kwan et al. [83] used a data set of about 50 million LiDAR points, but they do not mention the scalability of their work and how quickly it takes to process. I discuss this issue in depth in chapter 4.

- *Interactions* - There is a need for being able to analyze and manage LiDAR data in realtime. Otherwise, it is difficult to do comparative feature analysis or disseminate the results to responders. There is a need for interactive tools for analysis and searching to solve these problems. I discuss this in detail in chapter 5.

- *The lack of searchable content.* To be able to fully and effectively analyze the LiDAR data, the system must support: finding areas of similar elevation; finding buildings of similar size or type; finding road networks and their structures (e.g., bridges, embankments, ditches, etc.). In addition one should be able to find spatial relationships such as roads near areas of high vegetation; roads near coastal areas.

One hyptothetical use case could be that a severe hurricane hits a coastal area causing significant damage as well as changing the coastal landscape. Any previous LiDAR data may not be accurate, therefore new LiDAR would have to be collected and analyzed. Rescue teams would need to find appropriate areas for operations and safely landing helicopters to provide relief. In order to find such areas in newly mapped LiDAR data it would not be possible in real-time with current systems. To alleviate these challenges and make LiDAR analysis more interactive and effective, we need more effective visual analytics system that processes, analyzes, searches LiDAR data at scale (in a limited timeframe).

For understanding and analyzing remote sensing LiDAR data I utilized unsupervised learning. In order to do so, I had to define our feature space that best represented our data in order to perform unsupervised learning, specifically clustering.

I developed a process of taking a LiDAR point cloud and transforming it into a custom image with specific RGB values in which I have encoded our unique feature space. A table showing the analysis to RGB conversion can be seen in table 1. An overview of the analysis architecture can be seen in Figure 1. The first step of the process is taking the point cloud and generating a triangulated continuous mesh using Delaunay triangulation. I then compute normalized scalar values from 0.0 to 1.0 using these triangles through a specialized analytical process. I used prior work done in visually encoding unique attributes from

[123], composite density maps for multivariate trajectories. That work developed a flexible architecture for visually analyzing attributes that might reveal patterns. The authors were able to easily modify and calculate attributes and composite them into images in order to find unique patterns. In our work I propose three attributes that I believe were the most helpful in our analysis use cases for emergency response. These use cases focus on helping first responders, but the system is flexible enough that additional analysis techniques could be introduced or added to provide a new dimension of understanding of pattern analysis. Along a similar thread Kewei et al. [94] developed a statistical method for calculating a descriptor for stream-lines and vector fields in order to do clustering and comparisons. For our work I propose a method by which I vectorize certain properties of LiDAR data in order to create an image while encoding more information. Another approach by Thompson et al. [137] developed a new representation called hixels that provided a compact and information rich format, but added scalar-value uncertainty. I use pixels as our storage medium to provide a more compact form and easy to use with image analysis techniques, but in doing so it does provide some uncertainty by utilizing a scalar values.

I compute the normalized height of each vertex of each triangle based on its surrounding triangles. This allows for very easy comparison of a specific region of the resulting image with any other area in the image. Computing the normalized height is the hardest part of all the values I encode. Since I are normalizing the elevation to values between [0.0,1.0] it is very sensitive and difficult to quantify if there is a gradual slope in the terrain. For example suppose there are two buildings with the same height at two opposite ends on a LiDAR tile and there is a gradual slope in the terrain between them. Normalizing the elevation will result in one building being drastically higher then the other when both buildings are the same height.

Our work focuses on using just DSM LiDAR. I do not directly calculate the DTM (that is, the ground terrain surface without buildings or other features) from the LiDAR data to create the nDSM (normalized digital surface model, or the elevation of all objects on a flat surface). It is calculated by finding the difference between DTM and DSM. Doing so would require another computational step and require more memory. Instead I normalize as part of the analysis process. I looked at several approaches. The first was the normalizing approach described in the previous paragraph, but this produced undesirable results. As mentioned buildings of the same height were being miscalculated due to variances in overall elevation. The second approach was a brute force comparison using a defined radius to find nearby triangles, but due to the size of the data, on the order of millions, this operation was extremely slow and far from ideal. The third approach was sampling a large set of random triangles. This approach actually produced adequate results and was quite fast, but it did not always work.

I then explored using a quad tree and using the minimum and maximum of the leaf nodes in the tree. This technique proved to be optimal and provided enough computation speedup, but it did not handle cases of very flat areas. For example our data covers a coastal region with a wide mix of elevation types from water to downtown cities. The normalized elevation for the water areas was heavily miscalculated. To fix this issue I include one more step. To account for variances in gradual terrain slope I use a statistical approach to measure the variance compared to the minimum and maximum elevation of the entire LiDAR patch. I do this by first selecting a specific triangle and iterating to the lowest quadrant containing it in our quad tree. If the variance in elevation does not meet a specific deviation from the elevation of the entire LiDAR patch then I step up in the quad tree to the parent and do the calculation again until I eventually get to a quadrant that contains enough variance.

Table 1: LiDAR analysis table.

Color	**Analysis**
Red	Normalized Elevation
Green	Ground Slope
Blue	Estimated Roadnetwork

This iterative approach is similar to the method described in [48], but instead of iteratively using smaller and smaller windows I use the quad tree to find the best defined area for normalization.

The normalized height is the single most crucial value that I encode for doing any analysis. It also drives the rest of the values I encode. Since I select a region with enough statistical variance in elevation I use that region to calculate any comparison analysis that I encode. This value will eventually be stored as the red color channel in our final rendered image.

Surface Normal and Slope

The surface normal is the vector of the plane created by the triangle. I have to encode this three dimensional vector to a one dimensional value. I do this by taking a uniform directional vector and compute the dot product of it and the surface normal. This creates a one dimensional value between 0.0 and 1.0 that gives us the slope of the surface. This calculation is the same as computing the light intensity on a given surface using a directional light. The idea is that any surface with a steep slope with relation to the directional light will have lower intensity. Computing the slope of the surface allows us to very easily distinguish steep surfaces like buildings and trees. This value will eventually be stored as the green channel in our final rendered image.

Navigation Mesh and Drive-ability

During our initial work for determining values, I found it difficult to detect a road network using all the previous methods. Our initial analysis was for LiDAR patches in coastal regions. I found that it can be difficult to reliably detect roads from some types of LiDAR due to the nature of the elevation of the ground, very minimal building placement, and vegetation.

One option is to use existing road vector maps; while this would produce the best results it is not always feasible. The points in time when the LiDAR was collected and when the road vector maps were created can be different. Analyzing the differences between different periods in time, especially in rural and under-developed areas would require collecting older road vector maps and might not be possible in all cases. I wanted a consistent solution based on the LiDAR itself.

The solution I explored was generating a navigation mesh based on the geometry of the resulting triangulated LiDAR data [67]. I explored navigation mesh creation using an image-based approach with LiDAR [2], but the work was done in city landscapes and there is no mention of rural environments or any such results. In addition, most navigation mesh research aims to produce a simplified graph network for performing path planning. Our goal is not to produce an optimized graph, but to create an estimation metric for determining a road network. Therefore, I are not concerned about the complexity of the graph since I have no plans to perform path planning.

The first step operates on all the triangles independently in the 3D LiDAR data. It finds the best node size in the quad tree to perform the local normalized height. It calculates the local normalized height for that triangle, the dot product of the surface normal with a directional vector, the area of the triangle. Lastly it flags the triangle with a single bit value if it fits the criteria for being driveable. As part of the navigation mesh creation process I discard triangles with surface normals too steep for a road. I also discard triangles that are too small for a car to pass on.

I use the quad tree nodes to estimate a navigation mesh. Areas of high elevation change will most likely not have suitable geometry for connecting together into a navigation mesh, therefore I can minimize our search within those nodes and limit the amount of computation that has to be done. To compute an estimated road network I use a modified kernel density estimation using the quad tree as the range in which to do the local search and each triangle as a cell. In the local search I determine how many driveable triangles are connected with a particular triangle to form a continuous path.

This results in an estimated network that a car can physically drive on. While not completely accurate, it estimates a reliable road network that can be easily encoded into an image that is based on the LiDAR data itself. One of the problems of using this analysis technique is that it is very computationally intensive and would not be feasible in realistic time without considerations for scale.

SEMANTIC SEGMENTATION

One area of additional work in this domain is developing a pre-trained model with defined semantics. Using our analysis methods I can thus generate training examples to provide faster classification with meaning for raw triangulated LiDAR data. Using our analysis method I generated 6,926 labeled LiDAR images. This worked by using our interactive visual analytics tool to extract examples of buildings, roads, bridges, low vegetation, medium vegetation and high vegetation.

I then used those as labels alongside the raw triangulated LiDAR data. To increase our data sample size I applied horizontal and vertical mirroring to quadruple the training set to 27,704. For training I utilized a convolutional network for semantic segmentation (FCNs) [129]. Since our training dataset was still quite small I allocated 90% for training and 10% for external evaluation. Our model achieved 71.33% accuracy on the evaluation dataset. An example of the semantic segmentation on a new unseen triangulated LiDAR tile can be seen in Figure 2.

2.4 CROWD SOURCE

In this section I detail one significant type of crowd sourced data. While crowd source data can range significantly I present work focused on crowdsourcing geographical data specifically for indoor buildings and routing. Related work in this regard includes the OpenStreetMap system [56]. Crowdsourcing in OpenStreetMap was a successful means of utilizing public efforts to effectively map data that would otherwise be extremely expensive. This work was expanded to mapping indoor locations in OpenRoomMap [111], exploring the potential and accuracy of mapping buildings. However, the authors encountered several issues and the system restricted the floor plans to maintain accuracy. OpenRoomMap utilized manually modified architectural drawings that outlined every aspect of the building. Users interacting with the system were unable to make modifications or alter these outlines in any way. Additions to the building structure, like the modifications of rooms, would be impossible through the current implementation. Lastly, the system also encountered vandalism through outside user participation.

Models of 3D buildings are typically acquired from CAD drawings or designed using GIS systems and stored in databases. Most often, these tend to be in 2D with separate datasets for each floor of a building and additional attribute data held in different layers (structures). On the other hand, 3D interactive modeling has a long history, based on 3D interactive graphics systems and has been in common use in automobile, aerospace and numerous other engineering applications. Access to 3D building data has numerous applications, including indoor navigation. Visual information about a building can be very useful in the hands of users navigating an unfamiliar building, especially for first responders in an emergency or evacuation (e.g., due to the presence of a fire, chemical exposure, or a threatening and armed individual). The proliferation of mobile computing devices (e.g., smartphones, tablets) and the availability of powerful 3D graphics rendering capabilities that is currently possible on this platform opens up new opportunities and applications that were not possible even a few years ago.

Using crowd sourcing is an effective way to collect geo-spatial data [111, 56]. But, it presents significant challenges for evaluating user submitted data as well as scalability. Crowd sourcing requires a substainial amount of users to achieve effective results [41](Eaglin et al.). I discovered this during our user evaluations, which I discuss futher in chapter 6.

2.5 SOCIAL MEDIA

The use of social media has exploded in recent years generating massive amounts of complex data. This data has valuable uses for emergency response and critical situations. For example 20,000 tweets were generated each day during hurricane Sandy. 5,000 tweets were generated per second during the 2011 earthquake in Virginia [157]. Lazreg et al. [85] utilized deep learning as means of analyzing social media in crisis situations. Machine leaning is an extremely powerful tool to analyze and understand big data [97]. But, machine learning is also very complex, difficult to use and requires significant computational resources to work at scale. In this section I detail two major subtypes of data that makes up social media data as a whole text, images and videos. I then present novel work in the area of health with regard to social media.

2.5.1 TEXT

Text data is ubiqitous it can be found almost everywhere from print mediums like newspapers and books to digital mediums like blogs, websites and social media. With the advent of the internet and social media the amount of text data being produced has exploded. Effectively analyzing text data is difficult for several reasons. First, the data itself is so large it's impossible for analysts to read it in it's entirety. Secondly, text data is often unstructured. Lastly, text can be ambiguous due to language meaning, abreviations, miss-spellings, sarcasm, emojis and slang [49].

Twitter posts usually contain unstructured text consisting of 140 characters. This is the foundation of Twitter short and simple messages. Twitter can also contain links to additional content, images, videos, temporal data, and geolocation. Due to the inherent unstructured nature of Twitter messages work has focused on building better text analysis models for social media like Twitter [113].

Several new techniques have been developed in recent years to analyze text data. Topic modeling has become an effective tool for seperating textual documents into related groups using Latent Dirichlet Allocation [7, 154]. It works by creating a probalistic model based on a text corpus. The technique then uses a three-level hiearchical Bayesian model. This model then provides a probablistic measure of the topics for a given document.

Other analysis techniques include text clustering [134] and named entity extraction or NER [77]. NER is a powerful text analysis technique that extracts entities like time, people, places and organizations. It has downsides though. The models built for NER only apply to the languages the models are built on. Most models are also trained on a predefined corpus of text documents that contain correct grammar and spellings. Therefore, these models do not apply very well to unstructured documents like social media messages. Additional techniques have been developed using more advanced machine learning algorithms like neural nets. Kim et al. [76] used convolutional neural nets for sentence classification.

2.5.2 IMAGES AND VIDEO

Facebook posts can contain a plethora of different content. The content can range from short simple messages to very long and detail text. It can also contain temporal information, geolocation, images, videos, and links. In our own work we have examined the drinking behavior of college age students who post to Facebook. In that work I utilized machine learning to classify a range of different social media types by developing different models. These types included text, images and videos.

Images and more so videos are a unique type of data and often very challeneging to analyze and understand. Images are made up of pixels that are defined by numerical values for each given intensity, but the pixels often represent much more complex features like a persons face or object.

There are several different types of image analysis tasks like classification, segmentation and feature detection. More traditional analysis has focused on feature detection, segmentation, and blob extraction. Lowe et al. [93] developed SIFT, a technique for finding unique features within an image. Bay et al. [5] expanded on SIFT by improving the computational performance and detection. Image segmentation is used for extracting specific parts of an image based on pixel intensity or other types of features [108]. Mantz et al. [98] utilized marching squares for blob extraction. Blaschke et al. [6] utilized these image analysis techniques for detecting objects in images. Classification is a very important analysis technique for images and videos. Prior techniques have utilized Support Vector Machines [17] to perform classification tasks.

Deep learning is a very recent development in the machine learning community. It has provided significant improvement in these types of analysis tasks [124]. Deep learning relies heavily on GPU computing because the computational requirement is extremely high. In prior classification work features were manually defined by computer vision experts using techniques like SIFT [93]. Deep learning automatically learns features through an autoencoder [141]. Overall the process works by forward propagating labeled training images through the network. Then back propagation is performed in order to update the weights in the network. This is done through gradient descent in order to minimize error. Different types of solvers can be used based on the use-case and the type of data. Some of the well known solvers are Stochastic Gradient Descent, AdaDelta, Adaptive Gradient and Nesterov's Accelerated Gradient [104].

AlexNet [81] pioneered the use of deep networks, which has open the way for other networks like Google's inception network [136]. The use of deep learning has also been applied to other domains beyond images. Another image analysis task is captioning. This task is a crossroads between image analysis and natural language processing. This technique is extremely valuable because key words or captions can be generated that could then be used by natural language processing. Vinyals et al. [142] used multimodal recurrent neural networks to build a model trained on Flickr data to caption images. Karpathy et al. [69] developed a more recent image caption toolkit called NeuralTalk2.

Videos are more complicated than images. They share the same properties, but can consist of lots of frames. Videos also incorporate a time element so additional techniques have to be developed. Karpathy et al. [70] incorporated spatio-temporal information into a convolutional neural network. They trained their network on 1 million YouTube videos that consisted of 487 different classes. Joe et al. [105] used a convolutional neural network with several additions to classify videos. They incorporated a recurrent neural network that used long short-term cells connected to the final output of the convolutional neural network.

Table 2: This table details the distribution of data types gathered for training and testing.

Dataset	Attribute	Yes	No	Maybe
Training	Total	11.3%	79.1%	9.6%
	Text	3.6%	88.6%	7.8%
	Image	28.9%	60.1%	11.1%
	Video	15.2%	75.8%	9.1%
	Links	22.5	58.5%	19.0%
	Questions	0.0%	0.0%	100.0%
Testing	Total	22.3%	62.3%	14.3%
	Text	3.2%	87.3%	9.5%
	Image	28.2%	67.6%	4.2%
	Video	33.7%	59.0%	7.2%
	Links	20.8%	29.2%	50.0%
	Questions	0.0%	0.0%	0.0%

2.5.3 SOCIAL MEDIA FOR HEALTH

We applied these techniques to a new problem domain to identify health issues of college students with regards to binge drinking [45]. Over consumption of alchohol and underage drinking are very common on college campuses [46]. As a first step we developed methods for analyzing heterogenous social media data to identify drinking related content. As part of this work several different types of sources were collected and labeled from Facebook, Twitter, YouTube and Instagram. We focused specifically on text, images and videos from all of these sources as show in table 1.

We developed three unique classification models for each type. For text classification we utilized support vector machines. For images and videos I used deep learning and convolutional neural networks.

To construct our Support Vector Machine model for text classification we first removed links and hashtags in the text data as it is considered noise [62]. A feature vector is then constructed using the term frequency-inverse document frequency (TF-IDF)for each text post. We used a linear kernel for the construction of the SVM model.

For images and videos I produced two seperate models using convolutional neural networks, specifically (AlexNet). Since the data between the videos and images was signifi-

Table 3: Confusion Matrix

True/Prediction	**Yes**	**Maybe**	**No**	**Total**
Yes	38, 12, 26	7, 0, 0	15, 10, 2	15, 10, 2
Maybe	5, 17, 1	21, 0, 0	13, 7, 0	39, 24, 1
No	2, 25, 4	6, 0, 0	159, 23, 0	167, 48, 4
Total	45, 54, 31	34, 0, 0	187, 40, 2	266, 94, 33

cantly different I decided to produce two seperate models. From the 462 videos I extracted an image every 100 frames to use as a training set. Each image for both models is then resized to 256x256 in order to fit them correctly to the neural network. Nestrov's accelerated gradient was used for the back propagation phase of training.

To validate our models we used 5-fold cross-validation. The average accuracy of the text SVM model was 85.52%. On the withheld test data the model achieved an accuracy of 82.0%. The image classification model had an average accuracy of 72.0%, but on the independent test data it was less than 50.0%. This is heavily due to the training data sample size for images. In order to effectively train a convolutional neural network of this nature a significant amount of data is required. This can be seen in the video classification performance, since I had significantly more data, which achieved an average accuracy of 86.0% and 78.8% on the withheld data.

We then combined these models into one singular model whereby we classify based on the data type and accuracy of the model. For example if a post has both text and image data then we utilize the text based classification due to its higher accuracy. If a post has both text and video we utilize the video classifier over the text model. Overall the text model is the most reliable since majority (61.0%) of the training examples were text only.

2.6 SIMULATION

The last type of data covered in this work is simulation data. I focus on two types of simulation data the first is hurricane simulations and the second is evacuation simulations.

2.6.1 HURRICANE SIMULATION

Forecasting the destructive impact for a volatile hurricane on a network of vulnerable critical infrastructure network is a central challenge for emergency planners and responders in hurricane-prone areas. After witnessing the devastating destruction from Hurricane Sandy, decision makers in coastal US cities are on high-alert for threats to their critical infrastructures (e.g, power lines, food networks, shelters, etc.); they are requesting more robust simulation analyses to depict the potential impacts from another tropical storm.

Much prior research has focused on using simulations and predictive modeling to anticipate hurricane movement and suggest possible landfall and impact locations [140, 109, 44]. Due to the complexity and scalability of simulation runs, understanding these modeling efforts and their predictive capabilities from large collections of simulation results is challenging [122]. On the one hand, many of the traditional hurricane modeling approaches depend on a trial-and-error approach that is not always feasible for generating simulations consistently. While this type of approach is widely adopted, each simulation run requires analysts to fine-tune the parameters, which can be very time consuming and less methodological [37].

On the other hand, a new simulation approach is based on the idea of data-farming, which prepares many possible simulation outcomes in bulk [61]. However, this approach is largely limited by the simulation models, for which very few modelers have sufficient computing resources available to do sensitivity studies, validation and verification, effectiveness analysis, and related necessary activities. Exacerbating this challenge is that the large amount of such simulation results is far outpacing decision makers capability to analyze and make use of them.

To meet the challenges of dealing with disaster forecast and preparation, automated simulation methods are essential. However, they are not sufficient. There must be human input and direction, specifically for the interpretation of results and in some cases for directing the simulations. It is important for the decision makers to gain enough actionable knowledge such that they can respond in a timely and correct manner to possible failures of crucial infrastructure.

A key design component for our system is *Ensemble Visualization* of a large simulation space generated as a result of the aforementioned data-farming simulation approach. An ensemble, in this case, is a high-dimensional collection of attributes aggregated from raw simulation results that are centered around a single feature (e.g., Power Station, Airport, or Hospital).

Mittelstadt et al. [101] detailed visual analytics solutions for disaster management and focused on just the state of North Carolina. Expanding these simulations beyond North Carolina to Virgina, South Carolina, Georgia and Florida posed a significant challenge in data collection. For example a lot of the electrical grid was produced by hand. Therefore, the authors had to explore additional options for collecting that data. The other data sources collected were from FEMA Hazus , which provided all of the critical infrastructure and the FCC for all of the communication and cell tower data. Collecting and generating an up to date electrical grid network is not a trivial task and is very laborious to generate by hand. More recently work has been done in this area using crowd sourced data. Rivera et al. [114] developed a crowd sourced system called OpenGridMap to use in generating electrical grid simulations and models. Another system called SciGRID [100] used OpenStreetMap data to approximate an electrical grid network. I ended up using SciGRID because the electrical grid generated by the tool was already publicly available for the United States.

2.6.1.1 PATH PREDICTION

Since the ensemble simulations can take hours to days to run, it is not feasible to do so via a web-service. Therefore, I have developed several techniques for generating results that can be computed in a reasonable time. In order to reduce the computational complexity significantly I turned to machine learning.

Cases arise where a user would want to examine a new hurricane path that was not evaluated directly in our simulation space. Therefore, there is a need to develop an interactive real-time technique by which a user can draw a hypothetical hurricane path and receive analytical results derived from the prior simulations. I have developed a method by which I create a model using a SVM to classify hand drawn hurricane paths. Sencan et al. [127] employed support vector machines and random forests for classification of hurricanes tracks. Chorzepa et al. [22] used support vector machines to predict the vulnerability of coastal bridges during hurricane events. I trained the model to classify individual segments of the hurricane path in order to retrieve the best results. Each classified segment is then used to generate a spatial and path based query into our simulation results database. The aggregated queries are then used to piece together a complete result.

To build our path model I segmented 16 total paths into 29680 total training samples. This produced 1855 samples per path. I used 80% of the data for training and 20% for external evaluation. The model achieved 93.547% accuracy on the external evaluation dataset. Some of those paths can be seen in Figure 3.

Using the aggregated results I then have to align the temporal component of the outages. Since the simulations I run occur at completely different points in time (i.e. one hurricane might happen in the middle of August another the last week of September), there is a temporal alignment issue. This poses a significant problem when aggregating individual pieces into one continuous temporal signature. Therefore, I constructed a regression model using random forests [88] to roughly predict the outage time of a specific feature.

I perform training by utilizing every outage produced from each simulation run. For one hurricane path the simulation runs 1000 times. Each run produces variable results i.e. a specific hospital might be disabled at different points in time or might not be disabled at all. I use the relative simulation time in minutes and not the global date. Each of these unique occurrences are used for training. I construct a unique vector to identify each feature using properties like its latitude and longitude.

To construct our temporal regression model I utilized 6,017,322 different outage samples. I used 80% of the data for training and 20% for external evaluation. To measure the performance of our model I looked at several metrics: mean-square error, r^2 score, explained variance score, and median absolute error. The mean-squared error was 34134.00 (minutes). The r^2 score was 0.998. The explained variance score was 0.998. Lastly, the median absolute error was 65.989 (minutes).

2.6.2 EVACUATION SIMULATION

Evacuation of large urban structures, such as campus buildings, arenas, or stadiums, is of prime interest to emergency responders and planners. Although there is a large body of work on evacuation algorithms and their application, most of these methods are impractical to use in real-world scenarios (nonreal-time, for instance) or have difficulty handling scenarios with dynamically changing conditions. Our overall goal in this work is toward developing computer visualizations and real-time visual analytic tools for evacuations of large groups of buildings, and in the long term, integrate this with the street networks in the surrounding areas. A key aspect of our system is to provide situational awareness and decision support to first responders and emergency planners.

Macroscopic approaches focus primarily on minimizing egress time. The evacuees are treated as a unit or a group of units and moved from source to destination. Interaction among these units is defined by capacity/ congestion rules. Linear programming methods based on network flow were one of the earliest approaches that yielded optimal solutions, but at high algorithmic cost, making them impractical to use in real-world sce- narios; for example, the solution to the maximum flow problem has been implemented, with costs as high as n^3 yielding a best-known cost of O(nm lg n2=m). For each vertex, the sum of transit times of arcs on any path takes the same value, and for each vertex, the minimum cut is determined by the arcs incident to it whose tails are reachable. These assumptions resulted in a two-dimensional (2D) grid network, and they solved the transhipment prob- lem in O(n lg n) time. Shekhar and Yoo4 compare models relevant to the study of nearest neighbor paths. Also Kim et al.5 discuss contraflow in reconfigured networks for emergency route planning. In our work, this is relevant since dynamic changes to the building structure such as blockages or heavy congestion will require modification of the building network, followed by rerouting the affected occupants.

Our model is based on a heuristic approach, the Capacity Constrained Route Planner (CCRP) algorithm proposed by Shekhar et al. [128]. This approach attempts to find lower cost algorithmic solutions at the expense of the detail of each evacuees egress. These approaches are interesting because they can be evalu- ated quickly from a user perspective as the network flow problem is reduced to a generalized shortest path problem. The inputs are a graph of the building (or groups of buildings) and the evacuee population. The graph structure consists of nodes comprising various building elements (rooms, corridors, stairwells/elevators) represented as nodes and weighted edges (by distance) representing the relationship between the nodes (paths). The output is a route plan with start times, and a location matrix for each evacuee for each defined time segment. We have adapted this algorithm to meet the more challenging requirements for near real-time decision making in large urban environments, as well as the ability to inject situational changes during an emergency.

The route planner loads evacuation objects, which are combinations of urban structures, pathways, streets, and so on. Evacuations are built as combinations of structural objects and route planner objects and saved in the database. The route planner consists of the following components. Scenario construction. Scenario construction includes loading single- or multi-building evacuation objects, selecting a route planning algorithm (currently limited to our modified CCRP1) and setting visualization modes and evacuation parameters, such as building capacity, egress width, evacuee speed and density, and stairway resistance.

In our scenarios, we use an egress width of 6 ft. Building capacity is estimated as a percent of the maxi- mum classroom occupancy (summed up across all the classrooms in the building and known occupancies in other parts of the building). Classroom occupancies are known based on course schedules and can be used in real-world scenarios. Classroom occupancies are 3 ft2 per occupant plus 3 ft average separation between seats. Rooms less than 100 ft2 are assumed to have a single occupant (such as an office). The entire campus evacuation scenario uses actual room capacities taken from the course schedule. Evacuee speed is set at 3 ft/s,20 which is reasonable for urban structure building design considering a fire emergency. The maximum density is set at six occu- pants per 9 ft2, a bit on the conservative side. In our model, evacuees move only when the density within this space is less than this maximum, else they are stopped until the density falls below this threshold (when some of the occupants in front begin moving forward). We also include a parameter to induce occupants to choose exits on the current floor, as opposed to possi- ble exits on adjacent floors. This parameter is termed stairway resistance and has the effect of increasing the path length up the stairs to the adjacent floor. In our experiments, we set the stairway resistance to 10 times the actual stairway path length.

The second step involved graph simplification. The graph networks of large buildings alone can be complex. Expanding it to a campus type scenario can be overwhelmingly computational. Therefore, steps were taken to introduce a graph simplification procedure in which a large node based graph for a building was converted into a zone graph for different regions of the building. This reduced the overall graph complexity significantly and would allow the system to scale up to campus wide evacuations.

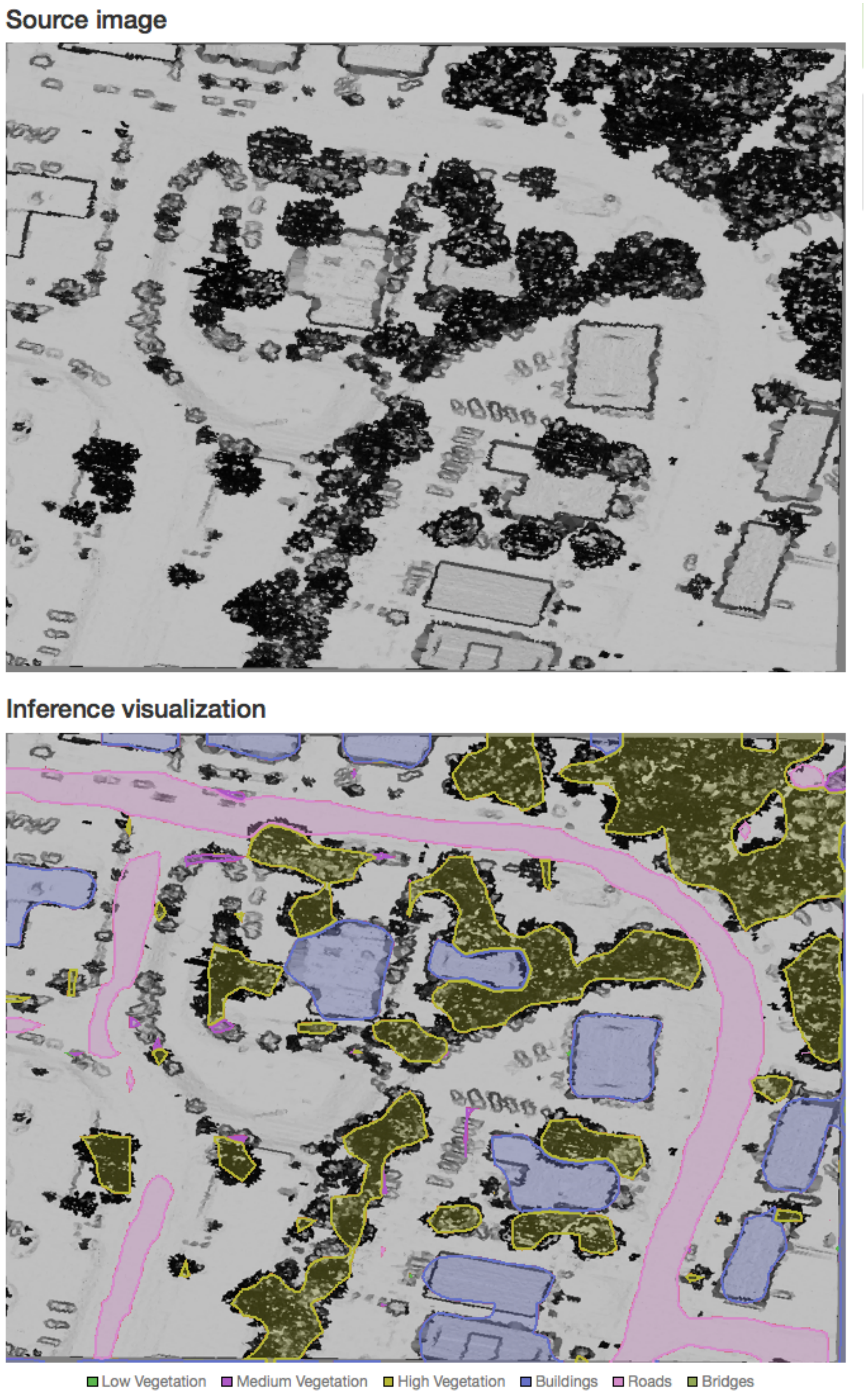


Figure 2: This figure shows an example of semantic segmentation applied to new LiDAR data.

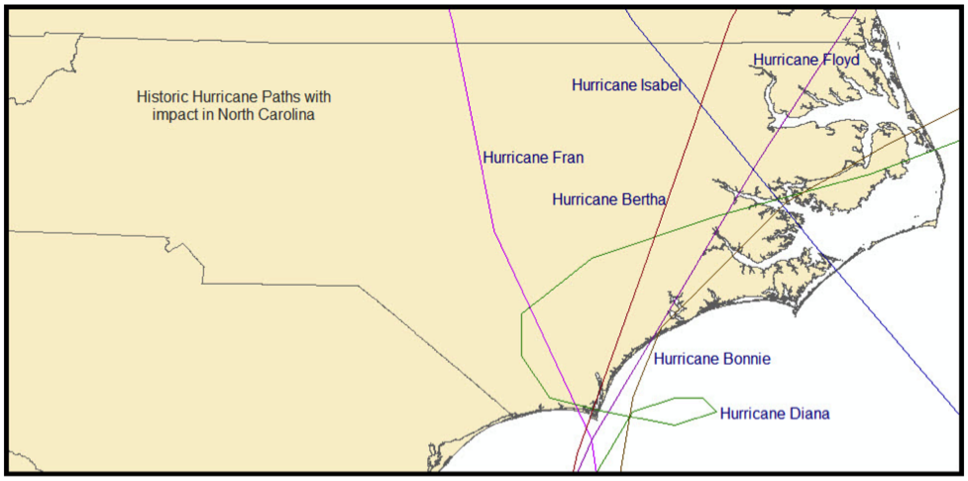


Figure 3: One of the models simulated with our critical infrastructure modeling. The image shows six Historical Hurricane Paths, including Bertha, Bonnie, Diana, Floyd, Fran and Isabel.

CHAPTER 3: VISUALIZATION

Visualization is an important aspect in understanding the value of big data. Without visualization it is impossible to discern meaning in order to make well informed decisions. The field of visualization is extensive and covers many disciplines. For conciseness in this disseration I focus on four domains in visualization that I have contributed to: spatio-temporal, hurricane simulations, remote sensing, evacuation simulations and emergency response.

3.1 SPATIO-TEMPORAL (STKDE) VISUALIZATION

Visualization is an important aspect for understanding spatio-temporal data to uncover hidden meaning. Space-time data has exploded substantially in the past few years making it a difficult challenge to visualize and extract meaning. As part of our contribution to spatio-temporal visual analytics I focus on one specific technique, the Space-Time Kernel Density Estimation (STKDE). Based on our results, I see the STKDE as an important addition that should be integrated with a larger set of tools for understanding and exploring space-time data [40]. In this section I discuss our contributions to the visual analysis of the STKDE. Ive incorporated our approach into a real-time VA tool with novel interlinked views. In chapter 4 I further discuss how I apply GPUs to solve the computationally complexity of the STKDE. In chapter 5 I discuss how I incorporated interaction for the STKDE into a web-based and mobile application. Lastly, I detail evaluation results of our user study in chapter 6.

Using a Kernel Density Estimation as part of a visual analytics tool is an effective way to understand geospatial hotspots [96, 95]. Since these density estimations are usually 2D and are typically applied to spatial extent, important temporal behavior or combined space-time behavior can remain hidden. Using a space-time kernel density estimation, it is possible to find clusters to reveal this behavior.

Kaya et al. [71] explored the usage of 3D and 2D on spatiotemporal data. In their evaluations, they examined the benefits and disadvantages of using 2D and 3D. For their visualizations they used heat maps for 2D and stacked heatmaps for 3D. They concluded that neither 2D nor 3D is significantly better for analyzing spatiotemporal data. But the use of stacked heatmaps in 3D does not take into account the relationship between different points in time, as does our approach. It just uses slices at different time intervals stacked together. In addition, it may be that the data to which they applied their analyses does not show the distinct space-time structure that I see in our 3D STKDE analysis. Nakaya et al. [103] explored the use of STKDE on visualizing crime clusters. Klemm et al. [78] explored the use of 3D heatmap visualization for population data. They found that the 3D visualization worked very well as an overview of the entire data. Clusters or areas of interest could be spotted quickly and from that point additional exploratory analysis could be pursued. Brundson et al. [10] explored the use of STKDE in visualizing crime patterns in space and time. Demsar et al. [33] studied using STKDE for trajectory data mining.

Gao et al. [53] used STKDE to visualize the impact of publications across space and time. In other work Gao et al. [52] explored STKDE compared to other space-time techniques and concluded that STKDE is much easier to visually identify clusters and areas of interest. Kapler et al. [68] developed GeoTime, another approach for visualizing space-time data. Kraak et al. [80] incorporated the space-time cube into geovisualization. Slingsby et al. [132] examined spatial classification of the UK population consisting of 41 different demographic types. Coordinated views were then connected with other visualizations along with treemaps that incorporated spatial ordering [150]. Wood et al. [149] examined spatial voting patterns in the London area.

3.1.1 3D AND 2D VIEW

In this section, I discuss the 3D and 2D view. I highlight the importance of our 3D view and how it is coordinated with other views. The 3D view is the most important view for visualizing our space-time kernel density estimation due to the 3D nature of the STKDE. Ive incorporated a 3D tile map with level of detail that you would see in other existing mapping systems. The 3D view has several common interactions. In chapter 5 I discuss in more depth novel contributions to how I adapted interactions for the STKDE to different platforms.

The first interaction is rotating. I utilize an orbiting camera fixed around a focal point. A user can rotate the camera around the focal point this is typical interaction you would find in a 3D application [19]. The second is panning, a user can pan the 3D view based on the viewing angle with relation to the up vector of the 3D scene. The third navigation interaction is zooming. Lastly, there is interaction with the 3D structures produced by the space-time kernel density estimation. These interactions include slicing, which is further discussed in chapter 5. The view is also linked with the other existing views, both the 2D spatial view, and the temporal view.

Using a 3D perspective introduces uncertainty and ambiguity into the visualization. Ambiguity can cause significant problems for users trying to use visual analytic systems. Examples of these ambiguities include trying to visually discern where different 3D features exist spatially and temporally in relation to one another. This is crucial for understanding orientation and position of 3D structures without significant cognitive load.

The first step I do is utilize a space-time cube around the entire STKDE. The lowest point on the Z axis corresponds to the oldest date and the highest point is the most recent. This technique was used in prior work and proved very effective for helping to discern the bounds both spatially and temporally of given 3D structures [30]. The space-time kernel density estimation does not generate any information about the center of clusters, but additional techniques can be used to find the high density centers. To minimize the spatial ambiguity I use the mean shift algorithm [26] in order to find high density centers.

This works by providing the algorithm only voxel centers with a high density. Once I find the high density centers I create a center line to the surface of the 3D map. For rendering the extracted surfaces I utilize phong shading with specular highlights in order to enhance and visually detail the surfaces. I also apply a slight transparency to the surfaces to help with occlusion. This can be seen in Figure 40.

3.1.2 TEMPORAL VIEW

The temporal view is the second most important view in our visual analytics tool. Therefore, I allocated a significant portion of the remaining screen space for the temporal graph. The temporal view depicts the aggregated timeline of all events that occurred across the entire data-set.

As part of our tool, I incorporated a coordinated time graph linked with the other views. Walker et al. [143] explored different time-series visualizations for effective interactions. I found the best choice to be a brush + zoom interaction and visualization. I provide a time graph that depicts the entire time-signature across the entire data set. A user brushes along this global time view to make a selection. Then a zoomed in portion of the time graph for that selection is created to show a more detailed signature of temporal events.

The time graph selection generates a corresponding bounding box in the 3D view that

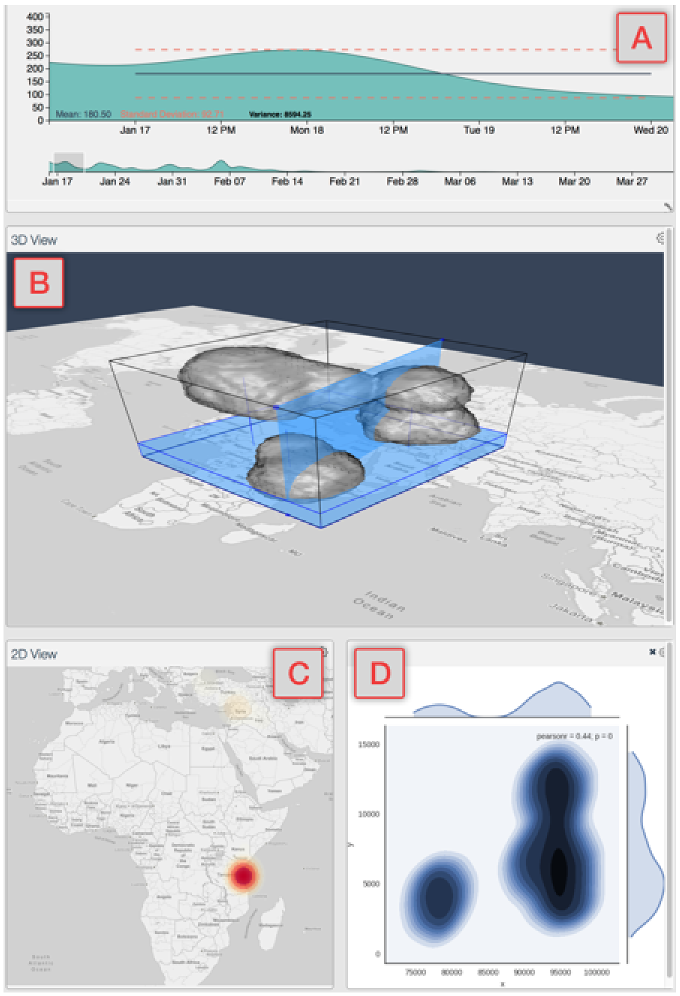


Figure 4: A comprehensive overview of our web-based tool using the STKDE. (A) 3D view with STKDE structures along with temporal and spatial slicing plane. (B) Temporal view with a selection made to create a temporal slicing plane. (C) 2D spatial view for viewing a temporal slice. (D) Spatial slice.

specifics the region in time that was selected. In the 2D view I generate a heatmap using just a kernel density estimation as show in Figure 40. This is discussed further in the section on slicing.

3.1.3 SURFACE EXTRACTION

As part of the visualization I use isosurface extraction to approximate the surface for a given density in order to create 3D structures. For the purposes of this work, I do not delve into this area because the field of isosurface extraction has been well documented. I use marching cubes to extract a triangle soup from the 3D density data. The triangles are then rendered on the client using phong shading with specular highlights.

3.1.4 VOLUMETRIC RENDERING

Volumetric rendering is an important visualization method for understanding 3D density data like the STKDE since it provides a detailed view of internal 3D structure (whereas an isosurface only presents 3D surface shape at a particular density value). An example of the STKDE volumetric structure is shown in Figure 31.

Volumetric rendering in most cases is a superior visualization over isosurface extraction. But, integrating volumetric rendering into a web-based tool let alone a mobile device in conjunction with numerous other visual elements can have adverse effects on user interaction. While mobile devices have improved significantly in GPU and rendering power, volumetric rendering or complex 3D rendering are still cumbersome because these currently platforms lack certain features and APIs to make this feasible. I discuss this challenge of scale in more depth in chapter 4.

I implemented a remote view-dependent volumetric rendering system. The volumetric visualization can be enabled through a menu option should the user wish to use it instead of the isosurface. When enabled, the STKDE density data is rendered on the server to an image overlay. The image is then sent back to the client. When the client navigates within the 3D view, the volumetric overlay is immediately hidden and the isosurface shown while navigating. This is done to maintain context plus interactivity while navigating. Once the user stops navigating, the view matrices are sent to the server; the volume is re-rendered and sent back to the client.

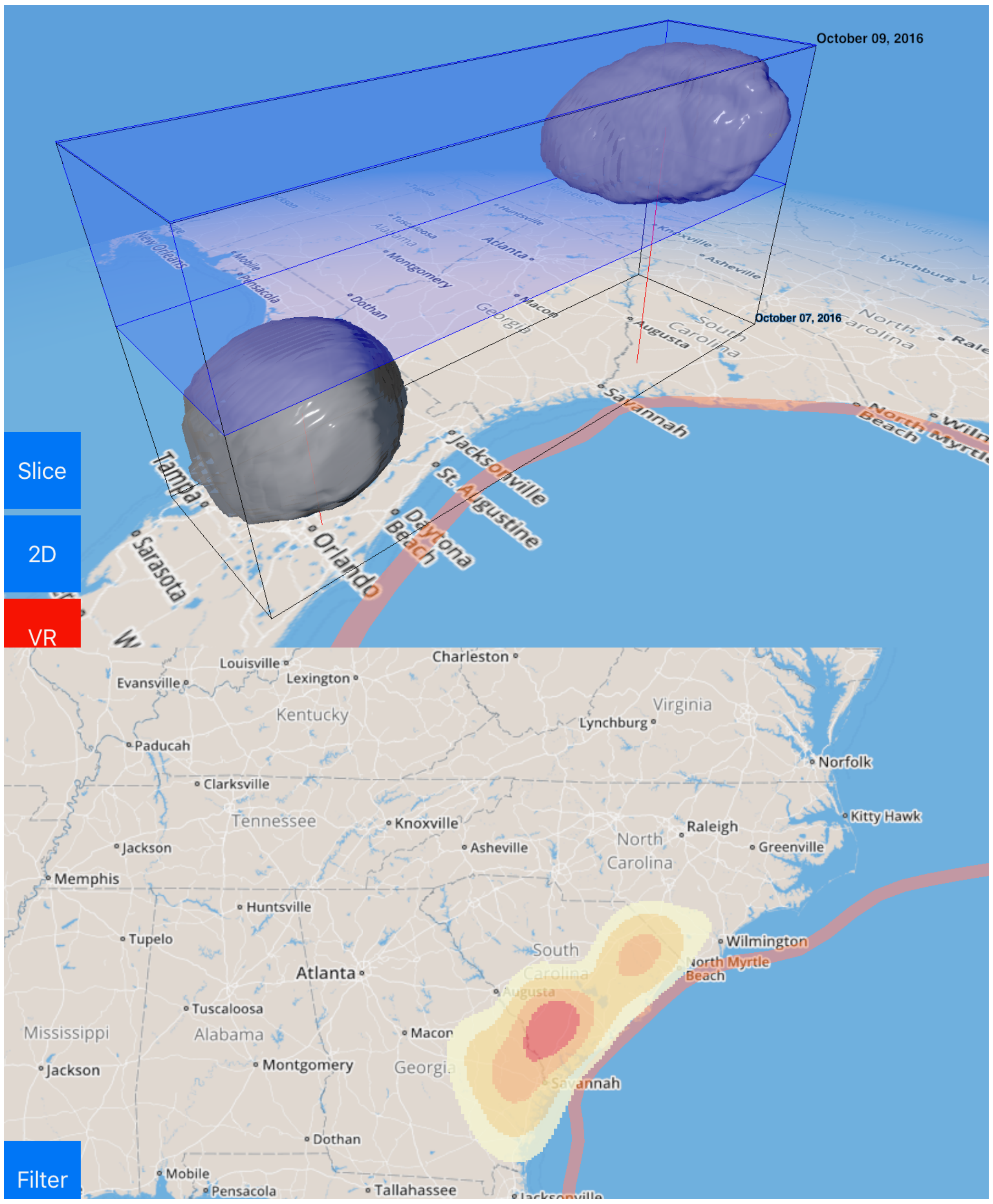


Figure 5: Top: An example of temporal slicing for a selected space-time volume. Bottom: Coordinated view of 2D heat map plus hotspots from the temporal slice.

3.1.5 SLICING

In this section I discuss the visualization portion of slicing. Slicing is an interactive visualization technique. Due to the complexity of the interaction different techniques have to be developed for different platforms i.e. web-based and mobile. In Chapter 5 I further discuss these interactions as they apply to different platforms.

There are two different types of slicing temporal and spatial slicing. I provide a time graph that depicts the entire time-signature across the entire data set. A user brushes along this global time view to make a selection. Then a zoomed in portion of the time graph for that selection is created to show a more detailed signature of temporal events.

The time graph selection generates a corresponding bounding box in the 3D view that specifics the region in time that was selected. In the 2D view I generate a heatmap using just a kernel density estimation as show in Figure 5.

In the user evaluation detailed in Chapter 6 I examined our tool with just temporal slicing. After doing the evaluation I realized that additional work had to be done for different types of slicing and that temporal slicing alone was not enough. Thus, from that evaluation I learned that I needed to add spatial slicing due to the unique structures created via the STKDE.

Spatial slicing is a new technique that arises from the STKDE but has some similarites to ThemeScan [106]. It is important because the STKDE volume is often not symmetric around its perpendicular axis. It may be stretched in one direction or have a bump; it may merge with another volume in a particular direction. The temporal behavior of a slice in one direction can thus be significantly different than that in another direction, and this difference can be meaningful, as I will discuss further in the use cases below. Therefore it is important to quickly and easily set up slicing so that volumetric features can be accessed

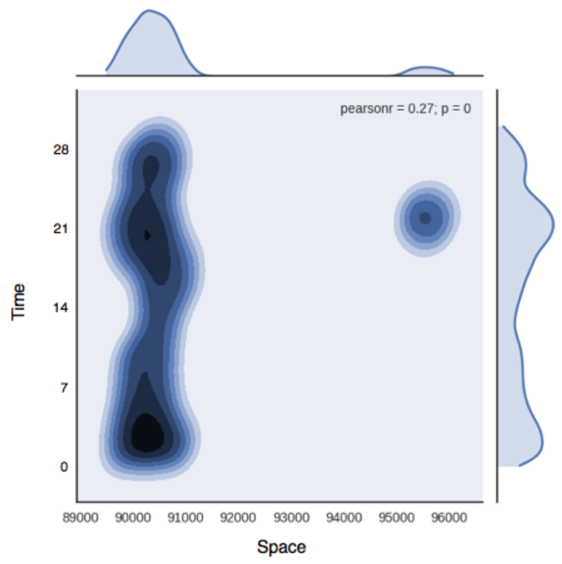


Figure 6: Analysis results of a spatial slice made for the Space-Time Kernel Density Estimation.

with reasonable precision and lack of ambiguity. In the same way that temporal slicing shows a distinct region across time, spatial slicing shows a slice in space.

I can provide two different types of visualizations for the resulting slice is shown in Figure 6. The contours provide a detail analysis. The user can see how they spread in the selected spatial direction over time and, in this case end abruptly. In addition, the image shows temporal and spatial components aggregated at the right side and stop, respectively. Thus, for example, the right side shows the timeline along the selected spatial direction.

Once the slicing plane is created, results are generated server-side and sent to the client. An interactive scatter plot is then generated from the results as seen in Figure 7. The X value of the scatter plot is determined by computing the spatial distance in miles of the specific feature i.e. hospital from the start point of the slicing plane. The Y axis is the elapsed time in minutes with respect to the first feature disabled. Thus the scatter plot shows the times when and locations where (in this case) outage events pile up. Figure 6 shows clearly the details of behavior within and between the two volumes of outages, including trends in time.

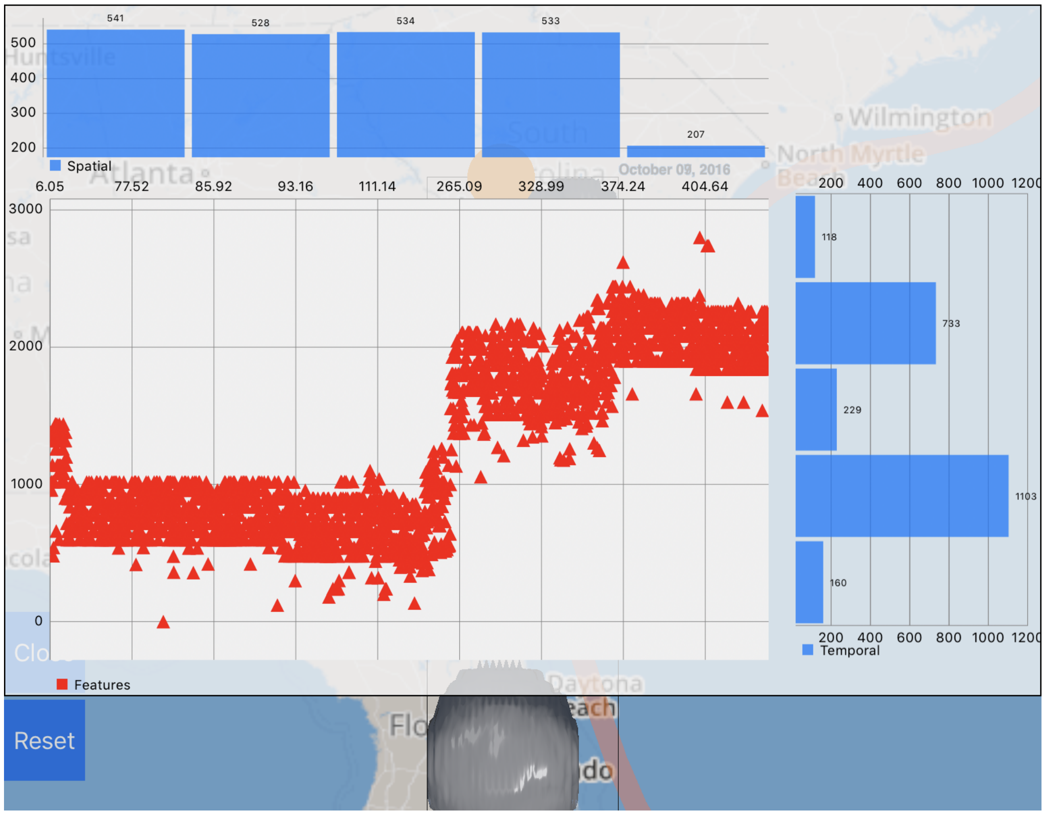


Figure 7: Spatial slicing scatter plot.

Note that the two STKDE volumes are essentially separated in time with small temporal overlap. The scatter plot results are binned spatially and temporally into a vertical and horizontal bar chart.

3.2 HURRICANE SIMULATIONS

As the first step in our ensemble visualization design, I followed prior visual analytics design studies [146, 126]. We also collaborated with first responders (UNCC Campus Police) to understand their workflows and analytics needs. We gathered this information through prior work on evacuation simulations [54]. Based on our discussions, it became clear that a key aspect of our effort was to provide a visual analytics system that can accommodate their in-field analytics needs. Since, I are focusing on a system with high mobility like a tablet device our visualizations and analysis tools must be simple enough to use with touch gestures and smaller screens.

Given our focus on the disaster response domain, I choose the geospatial view as the main entry view where all ensembles are presented at the initial stage. The coordinated views for temporal analysis and filtering are split into two transparent overlays. Specifically, the user is able to select either a filter view or a temporal view that contains a transparent overlay to see any changes made to the spatial representation. The user can also very quickly hide the overlays with a simple swipe gesture in order to return to spatial navigation

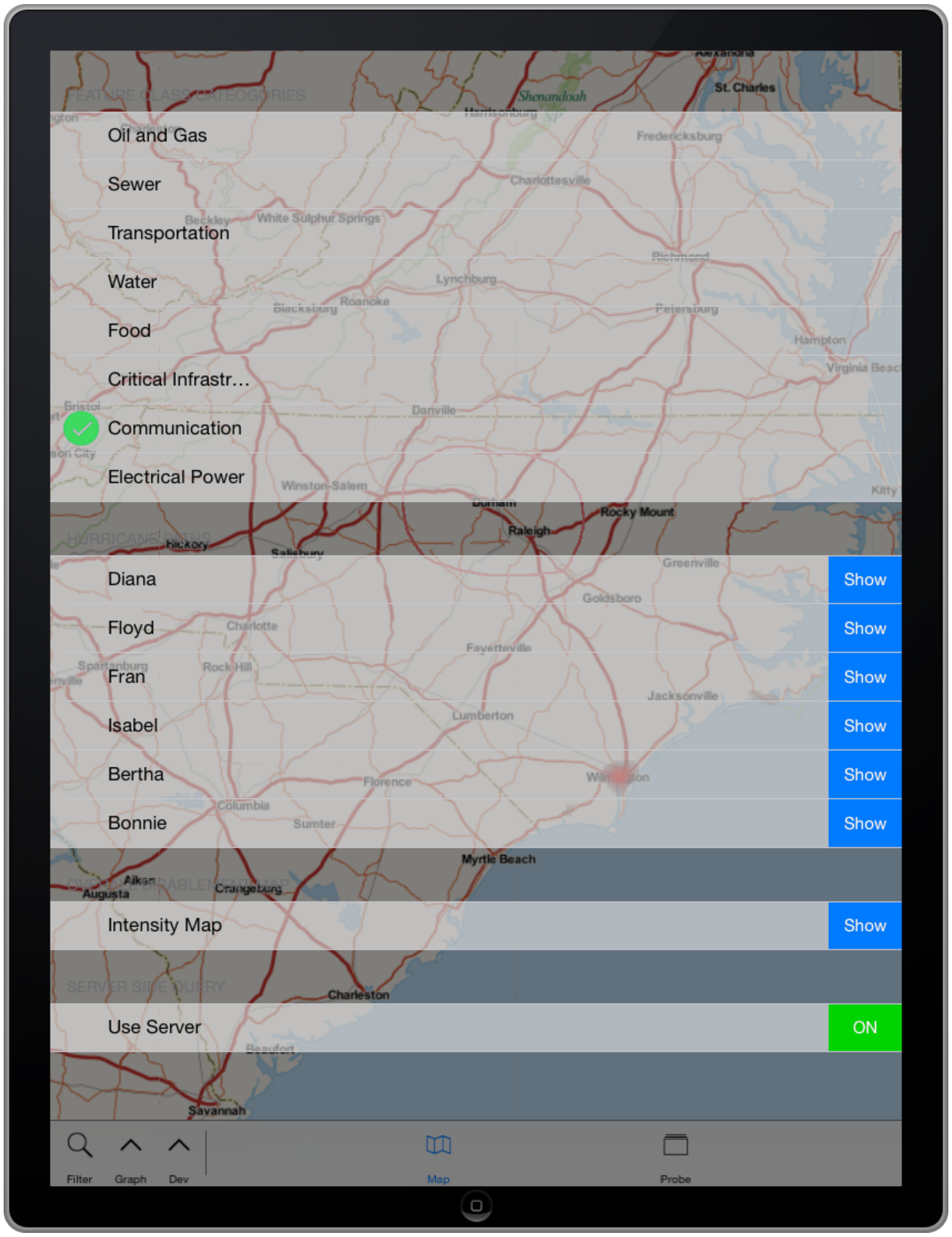
Our customized tile map system, as shown in Figure 40 (A), bears a similarity to commercial mapping systems, but also provide additional functionalities that fits our visual analytics needs. Since the map tiles are stored in the cloud and the visualization components are implemented as layers that can be stacked onto the map, our mobile interface remains lightweight. This enables us to completely customize the maps and it also allows us to embed certain features in the tile map textures to reduce the redundancy of rendering by having all the essential map details baked into pre-rendered tiles.

I provide two different visualizations. The first is a time-based color coding of each individual infrastructure based on the average time it went out. This allows first responders to see when specific disabling events occurred during a hurricane. The second visualization is a Kernel Density map. The intensity map shows the probability of outages for a particular region. This way users can quickly focus on regions with a high percentage of outages.

Understanding temporal behavior of a disaster is another important aspect for emergency planners to isolate time critical infrastructure. I have developed an interactive temporal analysis view that enables the selection of specific time ranges to depict what infrastructure was disabled as the hurricane passed through. As shown in Figure 8, the user can filter the simulation space in time and further examine the outages leading up to a peak or the peak itself.

Our system also enables the users to examine the result of cascading outages and interactively analyze the complex cascading relationship between different infrastructures. This allows the users to visually depict the interdependency of infrastructures and understand the effect of one infrastructure on others. Through the interactions, our collaborators (Managers from Duke Energy) were able to observe hurricanes with two distinct impacts patterns, namely the immediate sparse outages following high peaks of outages as well as the latency outage pattern where an infrastructure feature (e.g., hospital) will stay on with its backup generator for a longer period of time before becoming disabled. These are all key temporal factors for evacuation and planning efforts.

I have further setup animations to help the users more effectively analyze the outage patterns. Previous research has suggested animation plays an excellent role in revealing changes where cascading events take place [116]. This gives an overview of the events


FEATURE CLASS CATEOGORIES

Oil and Gas
Sewer
Transportation
Water
Food
Critical Infrastr…
✓ Communication
Electrical Power

HURRICANE PATHS

Diana	Show
Floyd	Show
Fran	Show
Isabel	Show
Bertha	Show
Bonnie	Show

OVERLAYABLE MAP

Intensity Map	Show

SERVER SIDE QUERY

Use Server	ON

Figure 8: Ensemble Simulation Filter View.

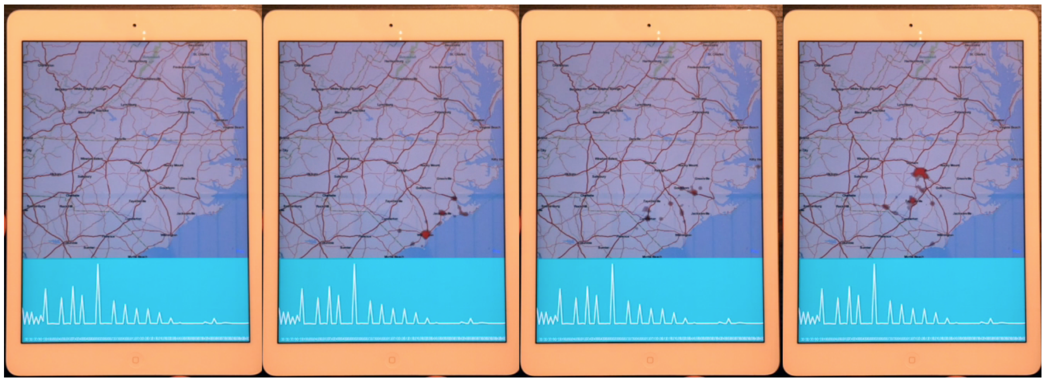


Figure 9: Ensemble Simulation Geospatial Animation.

that took place and hotspots during the simulation. However, enabling animation posed a computational challenge given the 50,000 features and millions of events I have in our simulation space. Computing each time segment for visualization exceeds the computation power of any existing commodity mobile device.

To address this challenge, I utilized streaming techniques that off-load much of the computation to our server and streams the visualizations in quick succession to a mobile device. As shown in Figure 9 shows several frames of the animated KDE, I have created animations of all the simulation events over a thirty day period using spatial KDE.

In addition, the same three data types: data points, geometry, and pre-rendered bitmaps that are available for single queries are also available for animations. A client can request all the raw data points across the entire animation span or the triangulated geometry or pre-rendered bitmaps. This allows the animation to be flexible on a wide range of client needs and scalable even down to mobile devices.

3.2.1 DETAILED VIEW

Each infrastructure in our simulation has a probabilistic chance that it will be disabled sometime across the entire simulation space. When the user zooms in close enough, I display a map glyph with the probability of each infrastructure as a pie chart, as shown in Figure 10.

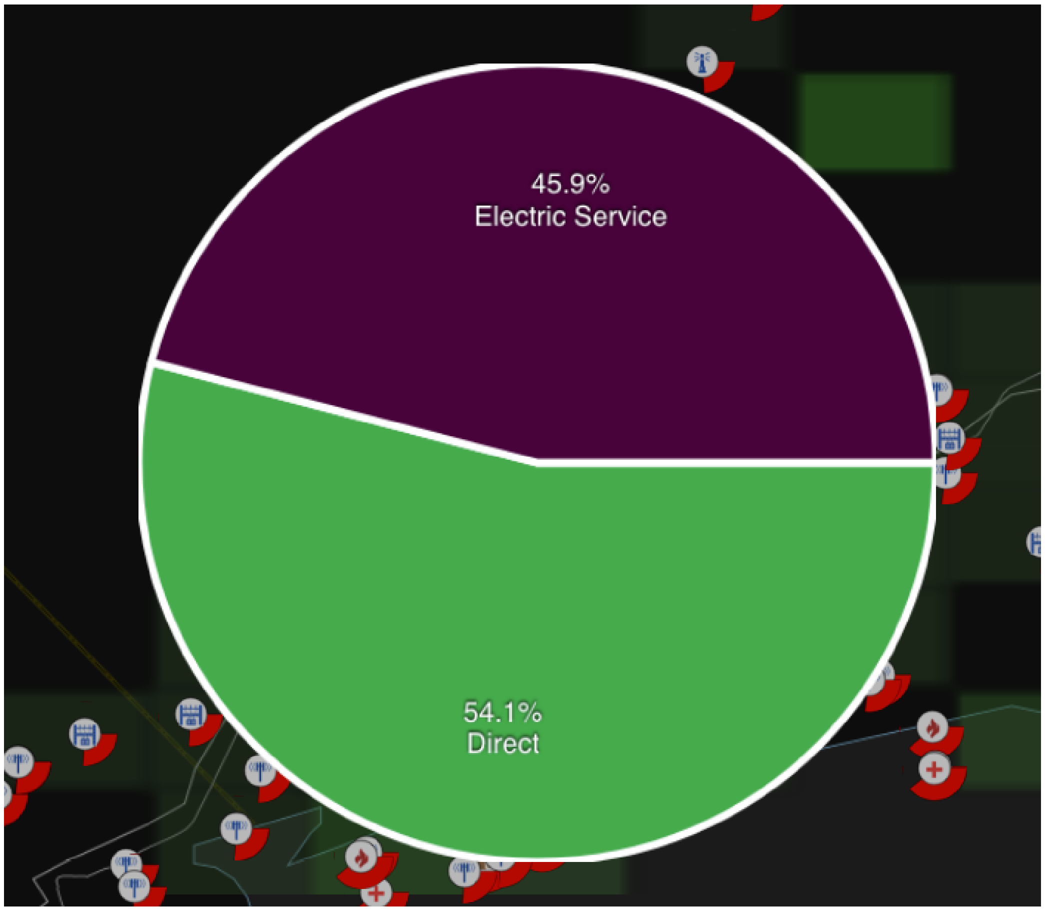


Figure 10: Ensemble Simulation Infrastructure Detailed Breakdown View.

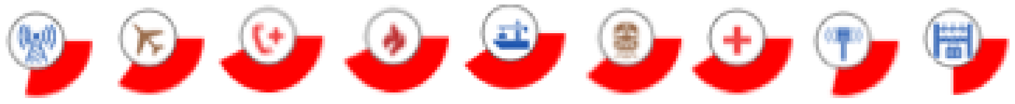


Figure 11: Ensemble Simulation Infrastructure Glyph Examples.

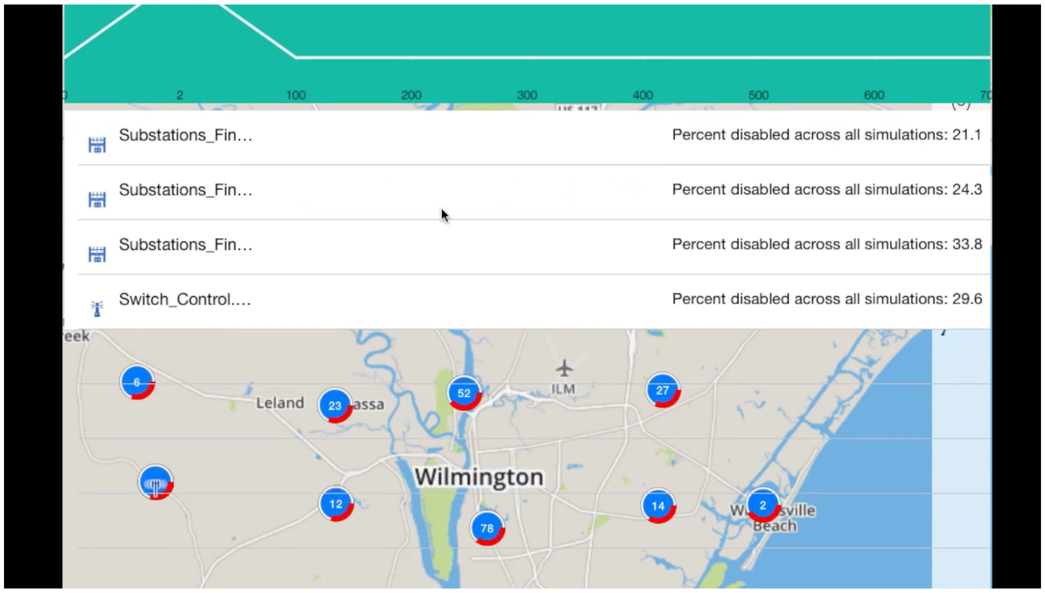
This allows the users to quickly see which infrastructure features have a higher probability of being disabled. More importantly each feature might be disabled from multiple events. By selecting an individual infrastructure a secondary pie chart will be displayed that shows individual causes for disablement broken down by percentage. In this way a user can isolate the main cause for the infrastructure being disabled.

The glyphs for each infrastructure are created on demand based on the type of infrastructure and the summation of its potential outage. Figure 11 shows a few examples of different types of glyphs. The inner icon is associated with the type of infrastructure. This allows users to easily discern what specific infrastructure they are looking at secondly, the outer ring displays the probabilistic chance the infrastructure feature is disabled. A full circle would indicate a 100 percent chance of being disabled. This outer ring allows users to quickly identify what features have the greatest likelihood of being disabled.

3.2.2 PROBING VIEW

Probing involves interactions that allow users to visualize a selected region in a free-form manner. As shown in Figure 9, a user can directly draw onto the map with his or her finger, drawing a bounding area around a region. This is an extremely important analysis tool because it allows the user to drive the analysis and focus on what is important to their geographic area. Anything within the bounding region will then be analyzed and returned to the user in a separate window.

When selected, a separate window will appear that initially shows the time distribution for all the features within the region. A user can quickly see any peaks in time in this


Substations_Fin... Percent disabled across all simulations: 21.1

Substations_Fin... Percent disabled across all simulations: 24.3

Substations_Fin... Percent disabled across all simulations: 33.8

Switch_Control.... Percent disabled across all simulations: 29.6

Figure 12: Ensemble Simulation Infrastructure breakdown view.

window. Secondly, the user can then view the breakdown of what feature classes are in the selection and how many were disabled (Figure 12). This allows users to see what features were disabled, how many were disabled and the time distribution of those features, permitting a very fast analysis of regions of interest. It also allows users to pinpoint features that have a high probability of being disabled. Most importantly it allows a user to compare multiple regions for joint analysis.

Each probe selected by the user is copied and saved for additional viewing, as shown in Figure 13. This way users can select multiple probes across many different areas and filters. Users can probe different time segments as well as different hurricane paths and commodities. Along with the selected features a screen shot of the region is attached to the data to give geospatial context to the region that was selected. All this information is presented in a secondary window where each probe is represented as a card with a map image of the selection and a time distribution of the events from the probe. When the user selects a specific card the view transitions to a detailed breakdown of that specific probe. In

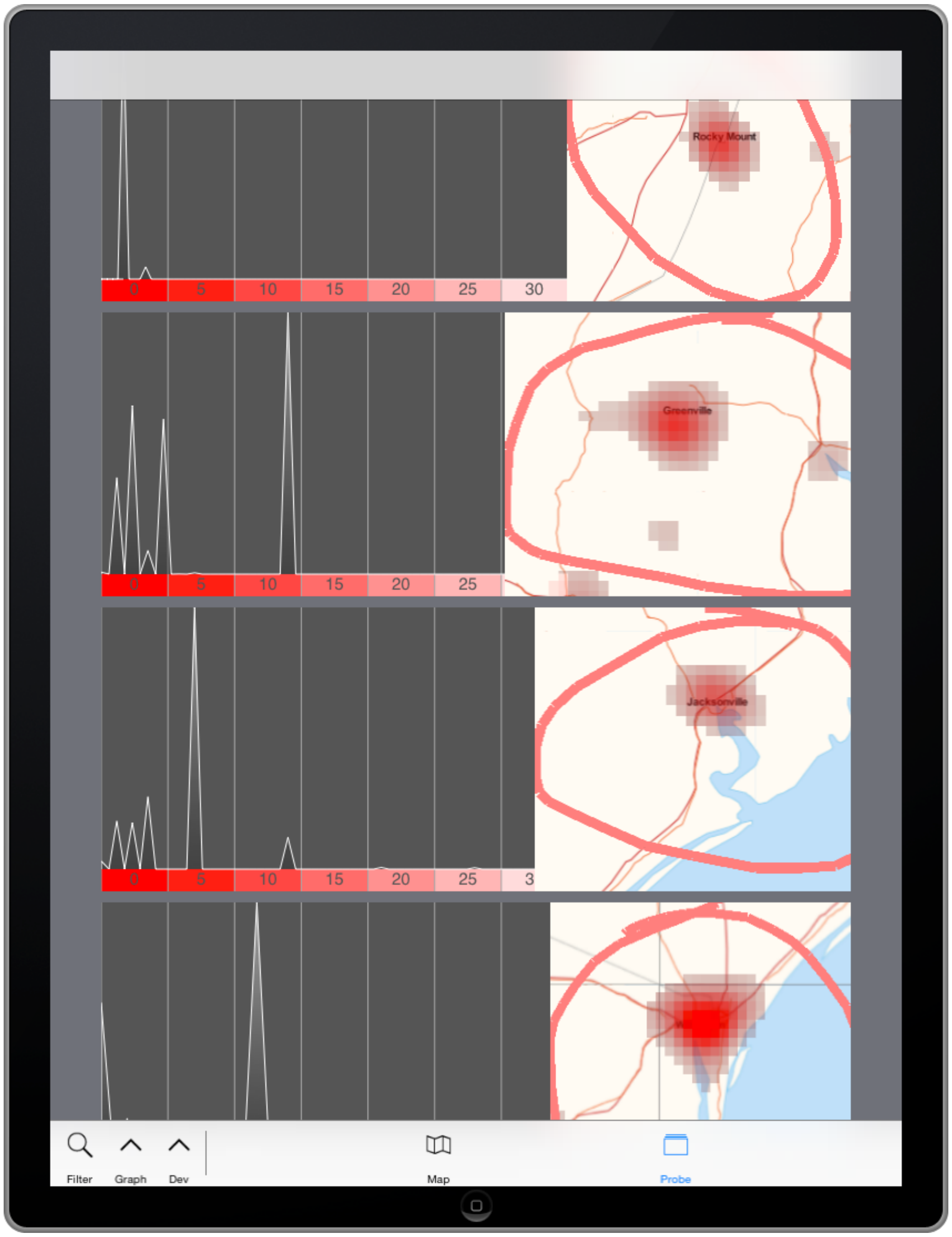


Figure 13: Ensemble Simulation Probe Card View.

this way users can return to further analyze individual events and time distributions from prior selections they have made.

3.3 CROWD SOURCE

In this section I detail the visualization aspect of our crowd source application to generate 3D buildings as shown in Figure 39.

I utilize multi-coordinated views for 2D and 3D navigation. The camera views allows users to switch between a top down orthographic view or a 3D perspective view with orbiting camera around a selected object. The mode type in switch allows users to change from creation mode or edit mode. Creation mode allows users to create objects like chairs, walls, doors, etc. The hidden menu on the top of the interface features a slidable menu that allows users to save their work, submit it or start over. The editing scene consists of a 3D tile environment in which users can select tiles to create new objects. Lastly the context options menu provides options based on the mode type. For creation mode it provides preset objects to make. For the edit mode it provides options for modifying objects.

3.4 REMOTE SENSING

In this section I describe the visual analytics interface that I have developed for understanding the analyzed LiDAR data. In order to do searching and finding I use thresholding and image segmentation. A user first makes a selection for some type of feature (for example a building) in order to do a search by example, an interactive technique. In Chapter 5 I detail this technique in depth, but the focus of this section is the visualization aspect.

As part of the user interface I present to the user two coordinated views as shown in Figure 15. One view is a global fixed view that provides a complete overview of all the LiDAR tiles. The second is a zoomed in detail view that allows a user to navigate around and zoom into specific areas. Together the views are connected. In the global view a bounding box represents the view bounds and position of the zoomed in view. Below the coordinated views is a scatter plot that displays all the similar feature results calculated from a selection in Figure 18.

For the scatter plot view I plot Along the X axis the similarity in width and height of specific feature. A value of 1.0 would mean the feature is approximately the same size. A value of 2.0 would mean the feature is approximately twice as large as the users selection and a value of 0.5 would mean the feature is about half the size of original selection. Along the Y axis I plot the histogram comparison. I compute this value by creating a color histogram for the original selection and all of the results found. I then use the function in Figure **??** derived from OpenCV to compute the correlation between two histograms. This metric ranges from 0.0 to 1.0 based on correlation, where 1.0 means the feature and selection have strongly similar histograms.

Within the scatter plot I display the max, mean, and minimum values from the resulting analysis. I also allow users to filter results further by sliding adjustable clipping lines that narrow results.

Any selection made in the scatter plot is then reflected in the detailed list view in Figure 20 and the global view in Figure 15. I incorporated glyphs to represent how similar a feature is to a users selection. The glyph is represented as a circular ring that is added on top of the global view. The more complete the ring is the higher the similarity exists between that feature and the selection as shown in Figure 15. A user can then further click on this glyph in the global view to reveal a detailed view of the feature in Figure 17.

I also incorporated a heatmap to quickly visualize hotspots as seen in Figure 16. I use a kernel density estimation that is weighted using the histogram similarity. In this way users can quickly focus on regions with the highest similarity of results. Lastly, I provide a concentrated list of the narrowed results from the scatter plot ranked by similarity in Figure 20. From this list users can select a specific feature that was found and visually examine it in Figure 17. They can locate it directly from the global view to find its origin. Also a user can create a new search using that feature to further refine what they are looking for. We followup with a usability evaluation, which we detail later in Chapter 6.

3.5 EMERGENCY RESPONSE

In this section I discuss our emergency response visualization application. The tool allowed first responders to navigate inside large urban buildings using a mobile device connected to a central command center. An example of the interface can be seen in Figure 22. The tool featured 2D and 3D visualizations of building layouts.

It showed glyphs for stair ways, exits, and elevators. Different colors were also used to denote rooms as pale yellow, hallways as light blue, stairways as orange, and exits as light green. Room numbers were also labeled. The tool also featured an integrated compass indicator on specially equipped devices shown in the top right. The tool also visualized routing paths that directed first responders to any location within the building. The routing paths could be seen in 2D and 3D.

3.6 EVACUATION SIMULATIONS

In this section I discuss the visualization reporting tools that were developed which allowed exploration into different parts of the evacuation simulation. Interactive bar graphs and time tables showed key points of congestion during the simulation which could be explored as to why those areas were congested. The system also reported automatically the points in time in which the most congestion occurred in-order to very quickly direct planners to those critical moments. An overview of the desktop application can be seen in Figure 23.

3.6.1 SINGLE-BUILDING VIEW

There are three components that make up the design. On the left is a 3D animated view of the urban structure, where the user can load and play evacuation simulations. Evacuees are represented as spheres and colored green, yellow, or red based on low to high congestion as seen in Figure 23. Partially transparent light green tubes represent stairwells, while purple cubes are exits. Blue polygons represent areas that can be occupied. Large red spheres of varying opacity represent edge (capacity) congestion. On the bottom left is the status tool widget, which allows moving around in the animation with a slider, indicating current step, evacuee counts, and total simulation time. The top right panel is the significant event window. The rectangular bars are menus with varying levels of detail of the simulation report. The bottom right panel is a scrolling widget with interactive charts and graphs for interacting with the simulation and visual analysis. Major features within the single-building view are as follows:

- *Congestion representation* - Congestion is a primary concern in evacuation scenarios and predictions of future congestion, and mitigation is important to emergency response teams. In our system, congestion levels range from green to red (low to high).

- *Temporal cues* - The color and size of spheres are modified at significant event times. For example, sphere size is enlarged when the simulator starts moving evacuees from a source location. The resultant pulsing in the animation informs the user of new sources of traffic and likely areas of future congestion.

- *Details on demand* - The significant event window in the reporting tool uses a colorized layered menu so that the visualization of significant information is presented as needed by the user. Significant events are evacuation events of interest to first responders in an evaluation of a given scenario. These event definitions came from discussions with University of North Carolina (UNC) Charlotte police.

- *Value* - The last part of Big data is identifying value from the data.

3.6.2 CAMPUS VIEW

Rather than show the evacuation of each building in its entirety, we show a more abstract view of the evacuation so that the incident commander can get a quick overview as seen in Figure 29. We thus use a 2.5D style visualization in the campus view. The building outlines have been extruded to show a 3D view of the campus; the campus buildings are viewed from above, and bars representing the total occupants in that part of the building are drawn at each time step. Each bar represents a 100 3 100-ft2 area, and occupants within that area are summed up across all the floors. In order to reduce clutter, only the areas con- taining significant numbers of occupants are displayed (using a predefined occupancy threshold). As congestion areas tend to be concentrated near stairwells (which form parts of exit routes), this strategy is reasonable; bars are colored relative to the size of the building occupancy, while the size of the bars are based on the total occupancy across the entire campus. Thus, it is possible to quickly understand the peak occupancies in each building as well as finding peak congested areas in the campus; occupancy ranges are mapped to a discrete set of colors, ranging from green (low) to red (high). Finally, selected areas represented by green cylinders are designated as staging areas for occupants as they exit the building. As the evacuation progresses, these cylinders grow taller, as a function of the number of occupants. Interactive querying for details of a part of the building or the staging areas will be supported in the final implementation, permitting the incident commander to obtain quantitative information during the emergency.

Thus, the campus view provides an overview of the evacuation for a large group of buildings; users can then select a particular building to bring up the building view (see Figure 23). Alternately, if the focus of the evacuation involves a few specific buildings, then these can be selected, and the detailed evacuation of the involved buildings can be analyzed. It is to be noted that the evacuation of all the buildings in the campus view runs concurrently, regardless of what buildings are selected in the current view; the building view simply shows a more detailed view of the occupants and associated geometry (corridors, stairwells, elevators, rooms, exits/entrances) at any instant.

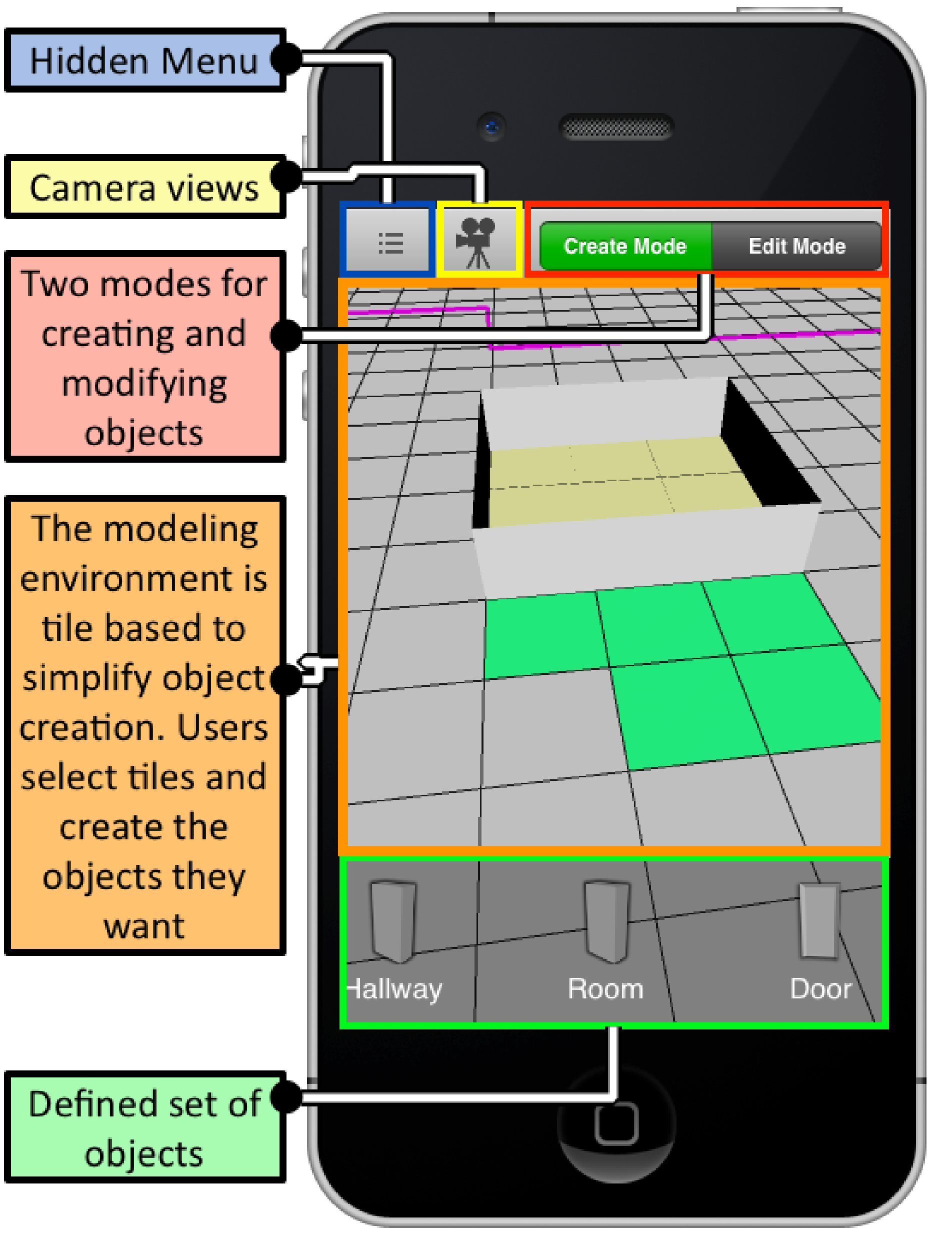

Hidden Menu
Camera views
Two modes for creating and modifying objects
The modeling environment is tile based to simplify object creation. Users select tiles and create the objects they want
Defined set of objects
Create Mode
Edit Mode
Hallway
Room
Door

Figure 14: Mobile interface for 3D Modeling By the Masses. This figure shows the interface layout developed to create 3D buildings from a mobile device.

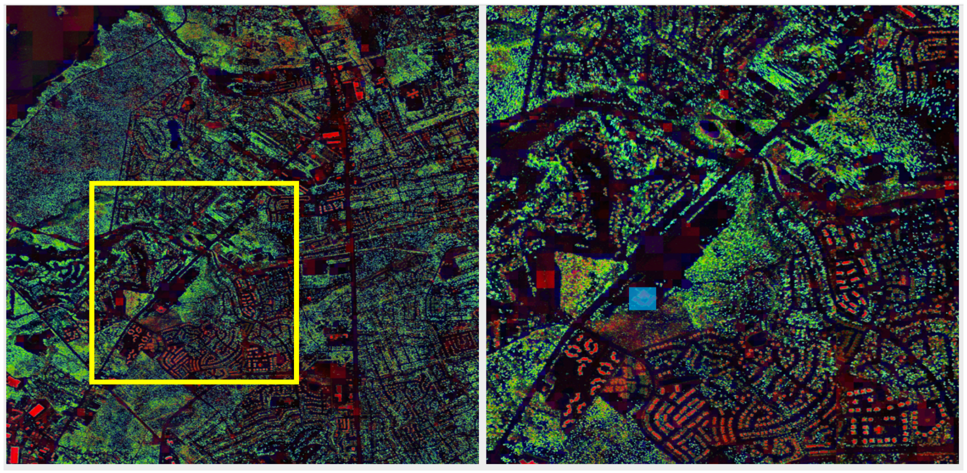


Figure 15: Two coordinated geo-spatial views. One shows an overview with a zoom-in region.

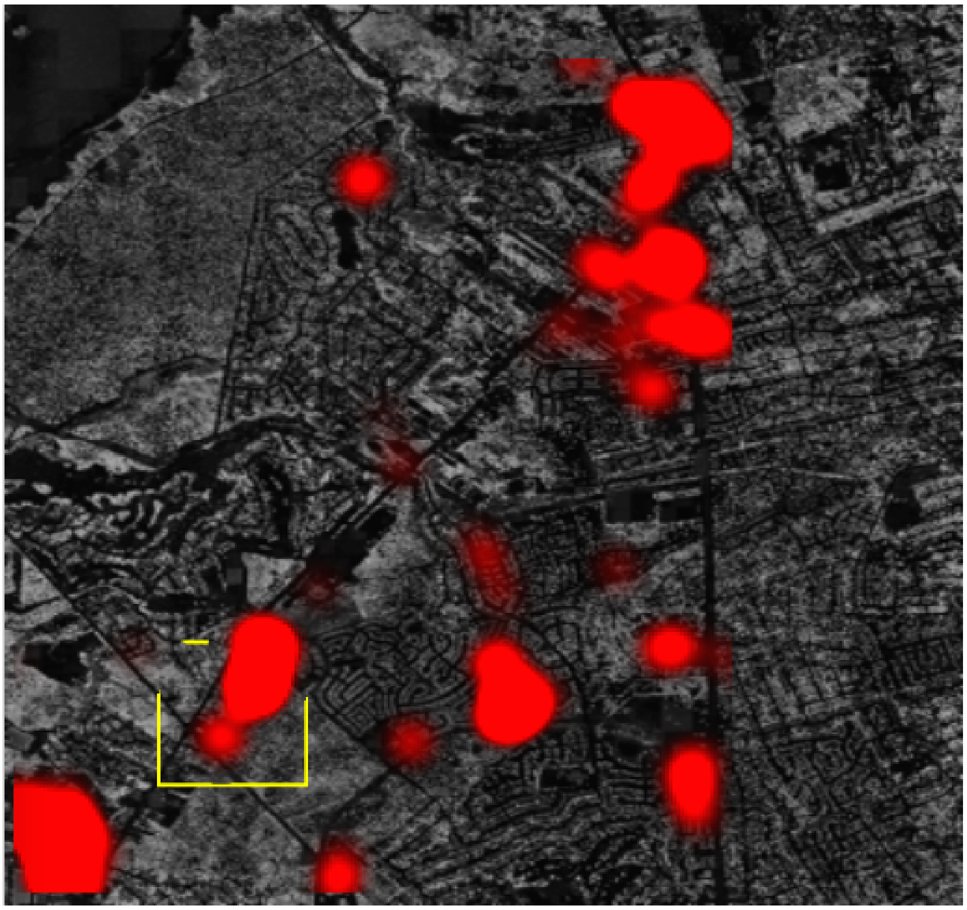
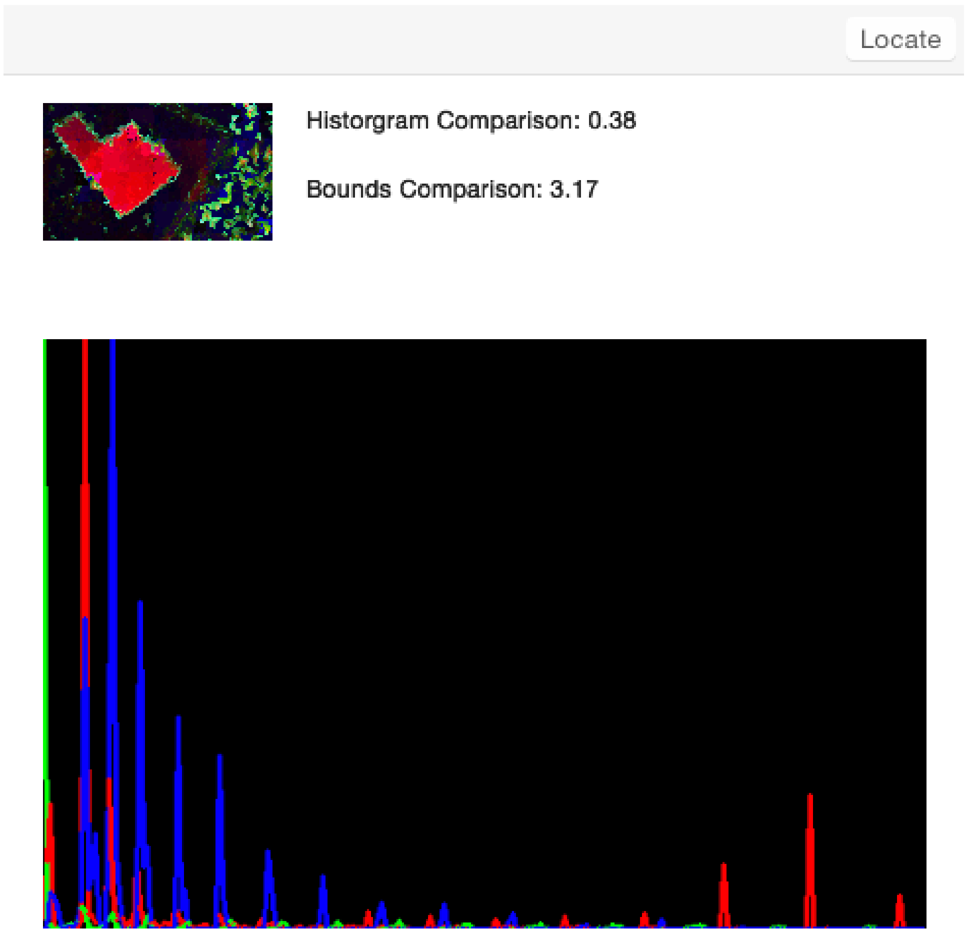


Figure 16: One of the coordinated views in our interactive visual analytics system for LiDAR. This view is the global view that shows a heatmap weighted by the similarity of the results. It also shows the bounding box for the the zoomed in detail view.



Historgram Comparison: 0.38

Bounds Comparison: 3.17

Figure 17: The detailed feature view, which shows the histogram comparison and the bounds comparison. Below it shows a detailed break down of the histogram for that feature. At the top is a locate button that allows a user to locate the feature directly in the LiDAR patches.

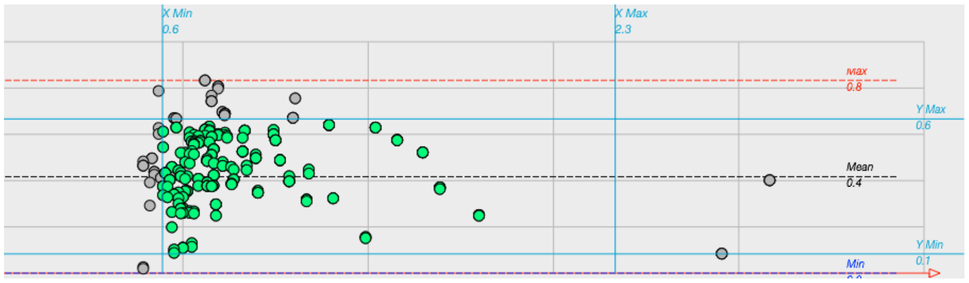


Figure 18: An example scatter plot showing a selected subset of data from a search. The subset of data selected is represented by the highlight green points. The points in grey are unselected and filtered out.

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \bar{H_1})(H_2(I) - \bar{H_2})}{\sqrt{\sum_I (H_1(I) - \bar{H_1})^2 \sum_I (H_2(I) - \bar{H_2})^2}}$$

Figure 19: An example scatter plot showing a selected subset of data from a search. The subset of data selected is represented by the highlight green points. The points in grey are unselected and filtered out.

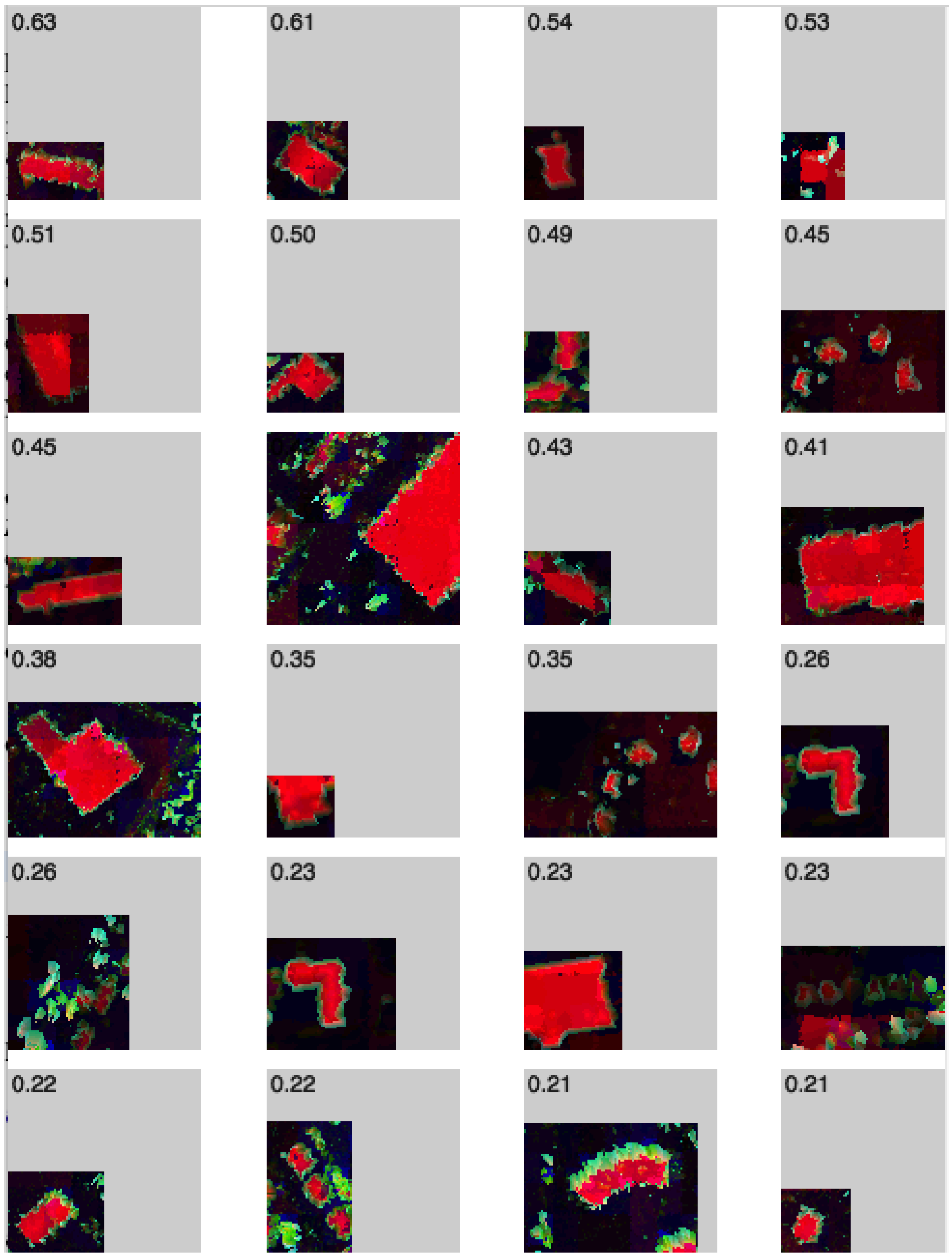


Figure 20: The detailed list view, which shows the sorted results of a search. Each cell contains a ranking and an image of the feature that was found.

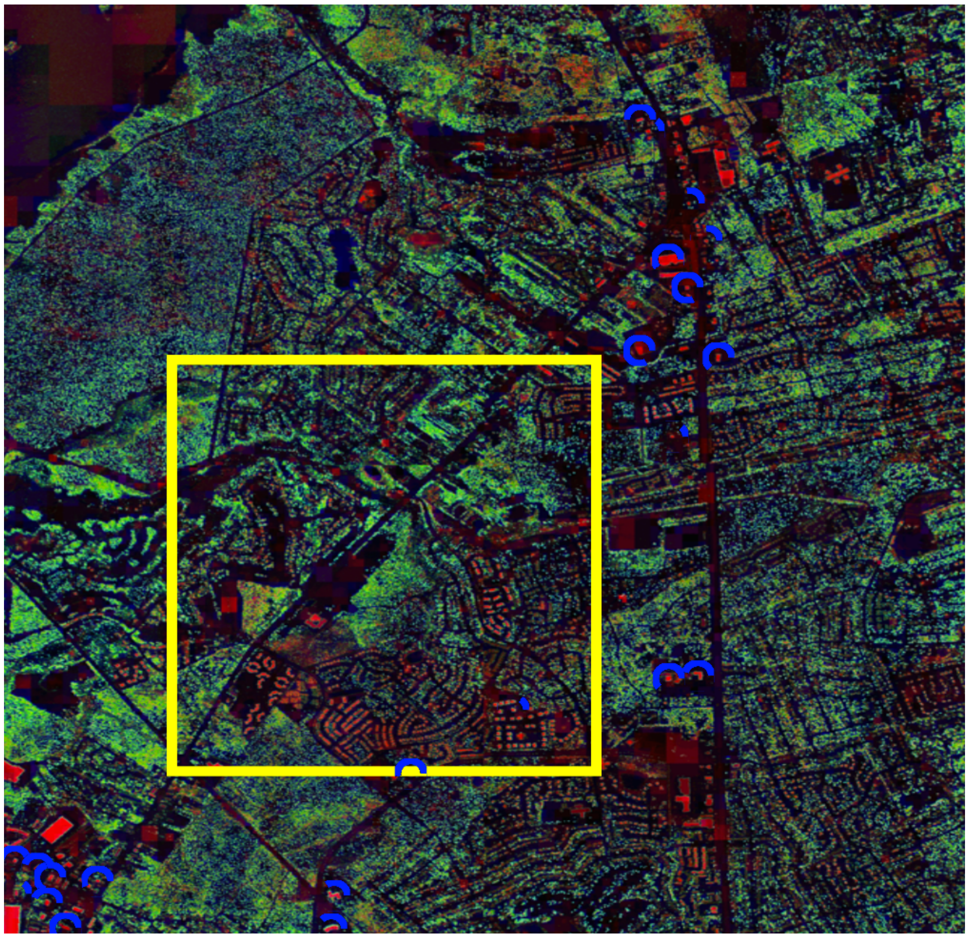


Figure 21: The global view showing annotatons that depict similarity between a selection and features the were found.

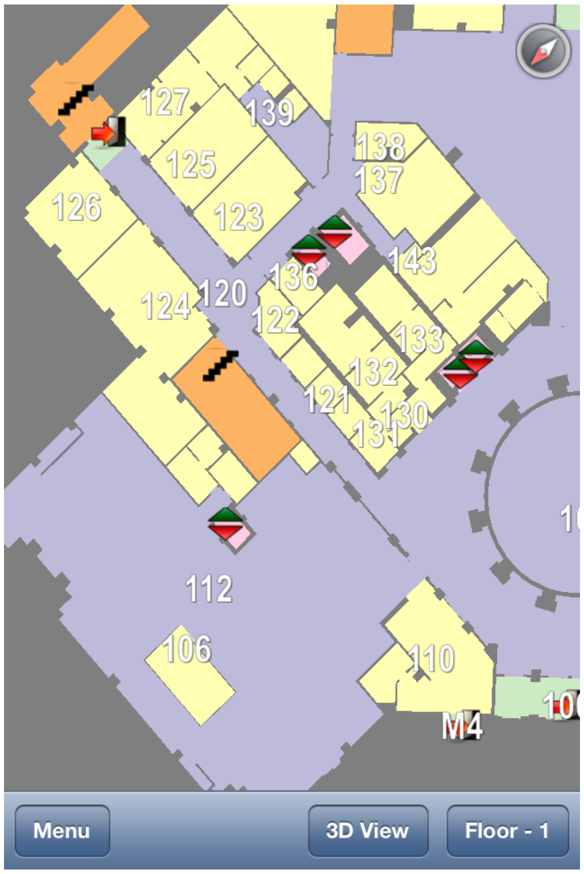


Figure 22: Interface for the Mobile Emergency Response system. The tool featured 2D and 3D visualizations of building layouts. It showed glyphs for stair ways, exits, and elevators. Different colors were also used to denote rooms as pale yellow, hallways as light blue, stairways as orange, and exits as light green. Room numbers were also labeled. The tool also featured an integrated compass indicator on specially equipped devices shown in the top right.

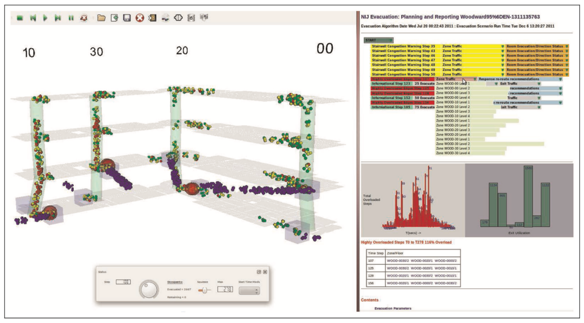


Figure 23: Interface for the Evacuation Simulation Desktop Application.

CHAPTER 4: SCALE

One of the significant challenges of big data is the sheer volume and variety of data that must be handled [9]. In the prior sections I have discussed the initial part of this challenge, which is understanding the variety of data and the techniques used for analyzing and visualizing those different types of data. In this section I detail the challenges for handling the scale of that data and our novel contributions to the field. I focus on the use of cloud computing to serve numerous types of clients, systems and platforms. The second major focus is the utilization of parallel computing. Since majority of the data and analytical computation requires siginifcant hardware resources it is not possible to perform these types of analysis at scale on mobile and commodity devices. Therefore, specialized algorithms have to be developed. I detail our contributions to distributed and GPU computing for accelerating data analysis and visualization.

4.1 CLOUD COMPUTING

In this section I cover cloud computing and client server architectures. The use of cloud computing is necessary not just due to the sheer volume of data and analytics, but also for all the access requirements by users, scientists and analysts. Techniques must be employed to provide scale and usability for all conditions and situations. Agrawal et al. [1] looked at the challeneges of cloud computing for big data. Due to the sheer volume and complexity of data, complex database management systems must be utilized.

The three most popular cloud computing paradigms are Infrastructure as a Service, Software as a Service and Platform as a Service. Traditionally, relational databases were used as the go to for data storgage, but they were designed for enterprise infrastructure. As a result they do not fulfill the needs of data management in the age of cloud computing.

Distributed databases systems [117] worked well for update intensive requirements. Parallel databases were established to feed intensive analytical applications [35, 36]. The purpose of parallel databases is to spread the load of data queries across multiple machines in order to handle demand. New data access patterns have also evolved in the age of cloud computing. Key-Value stores [14, 29] have become widely adopted for data access and caching. MapReduce [28] emerged for handling these data intensive tasks. HADOOP [139] an open source implementation has been widely adopted.

Another important aspect of cloud computing is caching and compression. Since data is transfered from the cloud to clients additional techniques must be incorporated for scaling up interactive data analytics. Caching is an important technique because accessing data from a hard drive for something like a database query can add considerable latency, especially if the operation is performed often. Therefore, it is important to cache data operations that are performed frequently, especially for analytical tasks that perform additional computation on the data. Dcache [155] is a data aware caching system for big data that uses MapReduce. Ji et al. [63] detailed the importance of caching in big data environments. Zhang et al. [153] surveyed the benefits of in-memory cache performance for managing and processing data for performance critical applications. Herodotou et al. [59] developed Starfish, a self-tuning system for data analytics. In their work they incorporated a caching layer to optimize their system and tools.

As part of the cloud computing paradigm and the internet of things data must be sent to client devices over a network usually the Internet. The gap between data size, consumption of data and the speed by which it can be sent has dramatically widen in recent years. Compression techniques to compensate for the I/O bottleneck are extremely important [50]. One of the well known and established compression algorithms is zlib [34]. Slow algorithms can negatively impact overall system performance. A higher compression ratio at the expense of speed can be beneficial due to the significant reduction in data that has to be transfered. LZ4 is one of the fastest compression algorithms and a good candidate to be used on-the-fly [75]. Other compression algorithms like LZFSE [39] which is several times faster than zlib and also provides higher energy efficiency. Liu et al. [90] incorporated compression as an essential aspect in their real-time visual querying system for big data.

In our work I have developed unique client-server architectures to handle these problems at scale. I incorporated theses techniques by utilizing in-memory caching for data analytical tasks and storing frequently repeated operations. Secondly, I have incorporated on-the-fly compression for large data requests like the STKDE.

4.1.1 CROWD SOURCE

In this section I detail advancements made in creating a client-server architecture for scaling up the crowd sourced collection of in door geometry (This work is also published in Eaglin et al. [41]). Developing this tool was a massive undertaking for both client and server. An overview of the architecture can be seen in Figure 24. The server handles all communication between clients, but its primary task is running an automated approval process. It works in several steps. First, users generate models using the mobile application. These are then uploaded to the system for approval and become pending models. These pending models then become public for other users to view and vote upon. Users then submit votes back to the server. As data is collected, the server continually analyzes votes for pending models. When a specific model passes approval, the server will combine that model into a newly approved model that will be viewable by the public. Users will then be

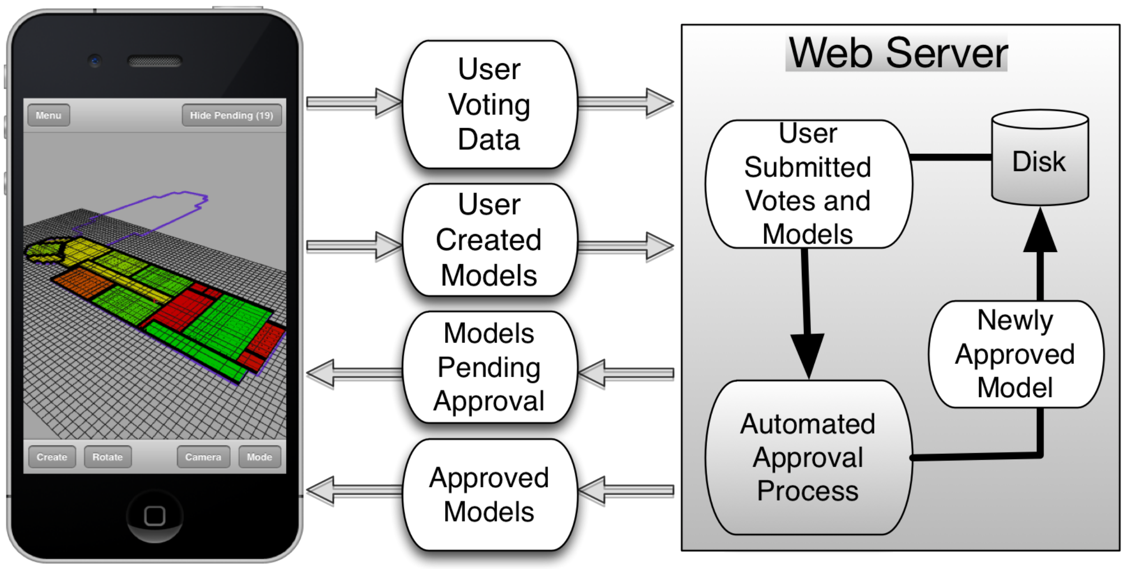


Figure 24: This figure shows the system architecture used for our crowd sourced collection and analysis tools for 3D Modeling By the Masses.

able to continue refining the approved model.

The mobile devices communicate over HTTP with an Apache webserver, which connects to a MySQL database. The webserver periodically runs the approval process, pulling new votes from the MySQL database. The server systems are written in PHP to send and receive data to the mobile devices. The geometric data sent from the server is compressed on the fly using gzip compression to maximize bandwidth.

4.1.2 HURRICANE SIMULATIONS

In this section I detail advancements made in creating a client-server architecture for our hurricane ensemble visual analysis tool (This work is also published in Eaglin et al. [43]). An overview of our system architecture can be seen in Figure 25. Our first optimization aims to accommodate the need to interactively visualize the entire ensemble space and its temporal changes. In this regard, I have employed a computing server that is designated to perform specific statistical calculations and data retrieval; these operations would otherwise be too time consuming on any mobile device.

Our computing server is connected to the database that contains all the raw simulation results. It handles data requests sent from the mobile device to the server, extracts the necessary data and finally computes the ensemble abstraction and encoding. I use a hash map for data transfer between the mobile device and the server. Each infrastructure has a unique key identifier. Using a hash map allows for quick lookups when doing additional computation on the mobile device.

Our mobile system is dependent upon the data pulled from the computing server. Any latency with this communication will affect the entire system. Due to this dependency I rely extensively on compression and aggressive caching to memory and disk. We cache to disk and memory on both the server and the client device, but majority of the caching takes place on the server. This limits potentially costly network requests and speeds up the mobile application.

I started with web services to query data on the fly, package the data and send it to the device. This method worked well with moderate datasets; as the data grew, however, the query time and packaging time began to slow down greatly. The response time had reached a certain threshold and at that point the application became unusable.

I therefore created a persistent environment instead of solely relying on a temporary service that queries the database each time and repackages it. This persistent environment caches the packaged data in main memory on the server. Querying the database can be improved through minor tweaks with caching and other features, but what substantially slowed down the entire process was that each time the web service is called a new instance is spun up in main memory to load the queried data in order to package it into a form usable by the mobile device.

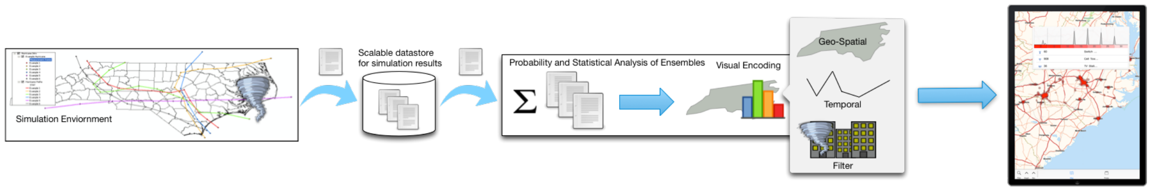

Simulation Enviornment
Scalable datastore for simulation results
Probability and Statistical Analysis of Ensembles
Visual Encoding
Geo-Spatial
Temporal
Filter

Figure 25: System architecture of our Ensemble Vis system.

I utilize parallelism for all our REST (Representational state transfer) requests to create asynchronous data pulls and provide three possible return types for scalability. Each type is specified by the client based on its needs and capabilities. The first is a list of all points returned from the query stored in a compressed JSON file. Returning all the points allows the client to run additional operations that might not be computationally constrained.

The second is a pre-computed kernel density estimation that is computed into indexed triangulated geometry and compressed into binary for transfer. This data type is for a client system that cannot calculate a spatial KDE in real-time time, but is capable of rendering the triangulated result. The triangulated result is resolution independent in order to provide a higher level of detail for visual analysis. The last type is a pre-rendered bitmap of the kernel density estimation. This is for client systems incapable of handling raw data points or even triangulated geometry.

4.2 PARALLEL/GPU COMPUTING

Li et al. [87] developed the PICKT acronym, a proposed solution for big data. The P in PICKT stands for parallel computing. They placed a major emphasis on parallel computing as a necessity for tackling big data. Jin et al. [64] examined the significant challenges of big data and proposed parallel computing as a solution for managing those challenges. In this section I focus on GPU computing as a growing area for handling large scale data analytical challenges. I discuss prior work in this area and novel advancements from our own work. GPU computing is a specialized type of parallel computing, that is optimized for floating point operations mainly used in vector and matrix math. The field has advanced significantly in recent years to accelerate computationally intensive tasks [107].

One area that benefits substaintially from GPU computing is machine learning. More recently data scientists have relied on cloud computing systems and thousands of CPU cores to train extremely large networks with over 1 billion parameters. Coates et al. [25] detailed the benefits of scaling up deep learning algorithms on GPU based systems. The authors used 3 GPU servers to train networks with 1 billion parameters in a couple of days. Raina et al. [110] showed the benefits of GPUs for use in unsupervised learning. Cireşan et al. [23] presented a high performance convolutional neural network for image classification. They implemented their technique using GPU computing in order to more quickly train their models on larger data sets.

Another area that benefits from GPU computing is graph analytics. Wu et al. [151] detailed a high-level programmable model for GPU-based graph analytics. Wang et al. [147] developed a high-performance graph library for GPU computing. In their evaluation they showed that for different graph analysis algorithms their framework has an order of magnitude speed up over existing tools. Wang et al. [144] developed exact triangle counting algorithms for GPU computing that provide the best performance over existing CPU and GPU implementations.

In our work I have applied GPU computing for LiDAR analysis as well as reducing the computation time of the Space-Time Kernel Density Estimation. I have also applied distributed computing to our hurricane simulations.

4.2.1 REMOTE SENSING

Zerger et al. [152] examined the current technical limitations with Geographic Information Systems (GIS) in emergency situations. They discovered that existing systems were unable to provide details in real-time due to limited processing power and the size of data. Richter et al. [112] examined a novel use of improving LiDAR visualization using GPU acceleration and out of core rendering.

During our work on LiDAR analysis, I incorporated GPU-based spatial-analysis techniques for analyzing massive LiDAR datasets [42]. In that work I used a quadtree with pruning and spatial indexing [57] to optimize data lookup on the GPU. On the GPU I are limited in several ways. The first is memory limitations, hardware limitations, and limitations in the programming API itself. That means I had to work around not having a deep call stack, no recursion, no pointers, limited local memory, and thread divergence. Therefore, I had to use an optimized data structure to limit memory and optimize lookups that was useable on the GPU.

As part of our implementation I process and load each patch independently. To do this efficiently, I must manage the data in parallel fashion using OpenCL/OpenGL and other high performance technologies. Ive placed all of our visual encoding operations onto the GPU. I begin by constructing the quad tree and spatial hash of all the triangles within the mesh. After I have completed this operation I pass these parameters into the GPU: triangle indices, vertex positions, vertex color, quad tree, hashed cells, and visualization parameters. Due to the synergy between GPU computation and rendering technologies I send all this information together because I can then instantly render the triangle data with the computed color data since it already exists on the GPU. This saves a significant amount of time due to the slow nature of copying data to and from system memory to GPU memory.

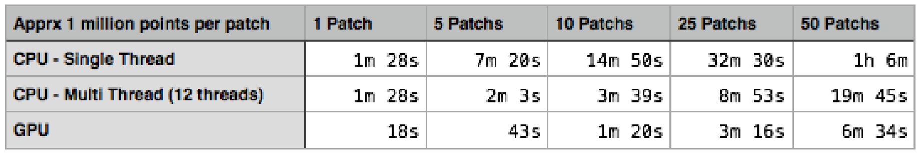
LiDAR data is complex and very large. So doing real-time interaction can be difficult. But, converting the data to an image and rendering it makes it easier to handle. Using images is also a great means of storing the resulting data after processing. In most cases the resulting image is dramatically smaller in memory footprint than the original LiDAR data used to produce it. I am essentially compressing the original data while adding even more information. To clarify we are strictly using point cloud data and not additional features like signal strength. The mesh generated is restricted to be a height field, therefore no concave structures like caves, natural arches will be included. The LiDAR analysis to RGB conversion can be referenced in table 1.

Using our data set I have multiple raw patches with four million points. The memory footprint of one of those patches is approximately 200MBs. Converting it to an image not only adds more information but dramatically decreases the size by almost a factor of one hundred to 2MBs through the use of lossless compression.

Since the image size is fixed that means it will scale very well. I ran three benchmarks using different core combinations to evaluate the scalability and performance of the algorithm for use in interactive applications in Figure 26. To tightly control the benchmark tests I used the same LiDAR patch, containing approximately one million points and had the application load it multiple times based on the number of patches being tested.

Once a set of LiDAR tiles have been converted to images they can be used to search using image analysis. Features that exist on the boundary of tiles will still be found in some regard, but they will be segmented and not represented as one complete feature. I use techniques that have already been proven very effective and are extremely fast. The first step of our process is using image segmentation through thresholding. This initial phase can be done

Computation Evaluation



Apprx 1 million points per patch	1 Patch	5 Patchs	10 Patchs	25 Patchs	50 Patchs
CPU - Single Thread	1m 28s	7m 20s	14m 50s	32m 30s	1h 6m
CPU - Multi Thread (12 threads)	1m 28s	2m 3s	3m 39s	8m 53s	19m 45s
GPU	18s	43s	1m 20s	3m 16s	6m 34s

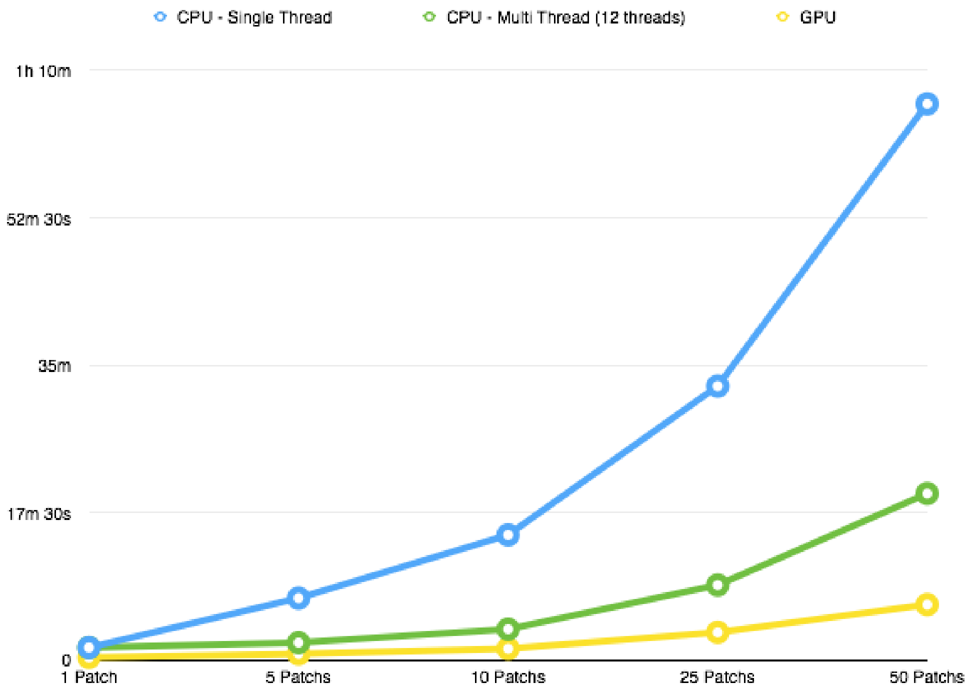


Figure 26: Timing results using three different metrics. The first is a single thread CPU implementation. The second is a multi-thread CPU implementation. Lastly, is our implementation with a GPU. The benchmark was performed on a 3.5 GHz 6-Core Intel Xeon E5, and a GTX 960 Nvidia GPU.

extremely quickly on a GPU and it instantly isolates regions that share similar values with a region a user is looking for. The entire goal of how I encoded attributes into images was for this step of thresholding. The process of encoding values into the RGB spectrum had to be meticulous because thresholding is very sensitive. The LiDAR analysis to RGB conversion can be referenced in table 1. The next step after thresholding is boundary analysis and blob extraction I also perform this on the GPU in order to achieve the up-most speed in searching. I perform blob extraction through marching squares [99]. The marching squares algorithm is embarrassingly parallel and fits well into our GPU workflow. Lastly I calculate an RGB histogram for each blob to further analyze later. Each histogram for each blob can also be calculated on the GPU depending on the size of the blob. I combine all of these steps into a continuous process whereby I keep as much information as possible on the GPU to minimize data transfer from host to device and vice versa. This allows us to very quickly find results from searches on very large datasets in realtime. Using a similar hardware setup for the encoding process and 1024 x 1024 image sizes I are able to search approximately 52 processed LiDAR patches per second.

4.2.2 STKDE

In this section I detail our contribution and advancement of scaling the STKDE technique both computationally and visually. Previous work in parallelizing the STKDE by Hohl et al. [60] utilized a parallel CPU approach. In their method they use a spatial decomposition initially to subdivide the workload among a defined set of CPUs. But even their approach does not provide enough speed-up in order to provide real-time analytics. I also compare our techniques to the most recent work by Lopez-Nova et al. [92]. In our work on analyzing LiDAR data on the GPU I utilized a quad-tree for graph based searching. These techniques were a starting point in optimizing our workflow for doing the STKDE on the GPU.

In prior work by Wang et al. [145], a decomposition step was performed to optimize the work load between processors using a quad tree for grid computing. Our parallel algorithm diverges significantly at this point. I bypass doing a decomposition step as it is not necessary using our data structure, and I operate at the voxel level to achieve the best parallelism instead of creating work groups. Our kernel function calculates the density of each voxel independently so as to utilize all GPU cores as much as possible. The hardest part of this calculation is collecting the necessary data points that fall within the space-time envelope used for a specific calculation. I use a threaded Oct-Tree to find the best nodes that fit the space and temporal bandwidth.

I store the Oct-Tree itself as a pre-order indexed array. The development of the Oct-Tree originated with the work done on the Quad-Tree in [42]. To reduce the memory footprint and lookup complexity I prune the tree depth. At a pre-defined depth based on the GPU memory limit, I stop subdividing and make a leaf node at the pre-defined depth. I take each leaf node to create a cell with an index for each point inside that leaf node. I store the cells in first-in order in an indexed array. The indices of the cells are used to create a cell range lookup. During the building process the cell ranges are stored at each node in the tree. After constructing the tree I then thread the tree. The purpose of threading the tree is that on the GPU I cannot use recursion nor is using a stack type data structure feasible. Since our tree is stored in a pre-order fashion I can iteratively step through the top level nodes of the tree quickly isolating sections of a query. As the algorithm steps through the tree it needs to know where to go next if a particular node does not fit within the space-time bandwidth or it does not contain any more children. This is the purpose of the thread. It is an index for the next location into the pre-ordered array of nodes.

I implemented our technique using OpenCL; in this way I could evaluate the performance of running not just on the GPU but on the CPU. Surprisingly, using our technique on the CPU achieved significant improvements in performance, but these improvements are nowhere close to what a single GPU can achieve. Later I also discuss in more detail why there is such significant improvement in the CPU performance against the current state of the art.

In order to make our approach more accessible and easy to use I wanted to incorporate it into a high level language like Python. Although Python has OpenCL packages I were unable to implement our algorithm due to the tight control necessary for our GPU-based data structures. Therefore, I opted to implement everything not related to the OpenCL computation into C++. This gave us the control I needed for managing memory and data structures to be sent to the GPU for processing. I then wrapped all of it into an easy to use Python binding. This allowed us to calculate the STKDE from a high level language, but retain all the control and performance from C++.

For the OpenCL computation I only use one kernel function. This function operates on each voxel in the space-time bounds. I found peculiarities between different device compilers. Although the OpenCL kernel is written in one language, it is compiled separately for Intel, AMD, and Nvidia devices. Due to this it is possible an OpenCL kernel could work flawlessly for one vendor and have issues for another. One interesting example is the use of the ternary operator. Using it in some cases had detrimental effects on our performance. As a solution I removed the ones I could and also switched to built-in OpenCL methods. I also stressed using the built-in OpenCL methods as they have better consistency between vendors. The pseudo-code of our algorithm can be seen in Figure 27.

In our implementation I automatically detect an optimal voxel size unless one is strictly given. I used the approach detailed in [131] to compute a optimal bandwidth. Due to hardware limitations I also limit the number of voxels based on the hardware configurations. For the hardware I used, I restricted the limit to 50 million voxels. This limit is adjustable; by either adding better GPU hardware or incorporating multiple GPUs it can be increased.

Weve placed a heavy emphasis on performance because all of our tools, interactions, visual analysis depend on how quickly I can calculate the STKDE. For all tests I used one CPU configuration consisting of an Intel Xeon with 6 cores at 3.5GHz. I also used a GTX 770 for the GPU benchmarks.

In the first part of our performance evaluation I examine our techniques against the state of the art [92]. In their work they evaluate a 3D KDE by generating two synthetic datasets with the mvrnorm function from the R framework, which generates multivariate samples with a normal distribution. I use the exact same approach to generate two sets of data with 500,000 samples and 1,000,000 samples each. I then vary the voxel count in relation to their evaluation tests. I also use the same heuristic approach to compute a bandwidth for the given datasets [28]. For 500,000 samples at 10 million voxels I were able to calculate it in 1.596 seconds on the CPU and 0.937 on the GPU. Using coalescing I were able to improve the time further to 0.74 seconds on the CPU and 0.679 on the GPU. For 1 million samples, in Figure 28, at 10 million voxels I were able to calculate it in 2.17 seconds on the CPU and 1.184 on the GPU. Using coalescing I were to achieve 0.9 seconds on the CPU and 0.727 seconds on the GPU. Compared to previous work I were able to achieve over 10 times an increase in performance.

I used a geomasked Dengue Fever dataset from prior work to evaluate our algorithm [60]. For our comparisons I used the same data, the same temporal, spatial and voxel size in order to maintain control on all these metrics. The data consisted of roughly 7,000 space-time points and the voxel count used was approximately 6.3 million. Running the STKDE sequentially takes approximately 23 minutes. In the prior algorithm a decomposition step must be performed before running the STKDE. For this setup the decomposition step took 62.96 seconds. I include this in the timing metrics because it still adds a significant amount of processing time. For our test I evaluated 6.3 million voxels. The prior algorithm took approximately 245.86 seconds at 6.3M voxels to execute the STKDE running on an Intel Xeon with 8 cores at 2.97GHz. Running our algorithm on the CPU took approximately 3.425 seconds. This is about 72 times faster using 2 fewer CPU cores. This is a significant improvement, but pales in comparison to the GPU. On the GPU I were able to achieve the same calculation in 0.636 seconds. This is almost 386 times faster. Overall I are able to achieve sub-second computation times in order to produce real-time results for voxel sizes that can be transferred through our web- based services.

4.2.2.1 SURFACE EXTRACTION

I realized that no matter how long the Space-Time Kernel Density Estimation took if the isosurface extraction took a long time as well it would be detrimental to our work and would hinder using the technique in real-time analytics. Therefore I opted to using a parallel GPU-based marching cubes to help improve computation time. Our efforts in this area could be further improved by more recent work that could be incorporated into our systems. Smistad et al. [133] developed a real-time technique for surface extraction that uses GPU computing and histogram-pyramids to further optimize surface extraction.

As an additional improvement I utilize the synergy of sharing GPU resources between operations. I do this because the raw data generated by the STKDE can be on the order of millions. Copying it from the GPU to main memory and then back to GPU memory in order to extract the isosurface is an extremely unnecessary step.

4.2.2.2 VOLUMETRIC RENDERING

Although commodity devices like tablets and laptops have improved significantly in computational and rendering power, volumetric rendering is still difficult to achieve at a reasonable frame-rate, especially since it needs to be integrated into an interactive, multi-window environment. A solution involving sending the entire 3D density data to the mobile device as part of a web-service is not feasible, as the data could be too large for cellular connection.

There currently exist limitations in rendering APIs for web-based tools. WebGL, currently the most capable 3D graphics API lacks support for 3D textures, which is a fundamental need for volumetric rendering. Work-arounds are possible but they provide sub-optimial performance for an interactive application.

Shi et al. [130] developed a remote real-time rendering system specifically for mobile applications. Lamberti et al. [84] proposed streaming-based 3D visualization for mobile devices. Krone et al. [82] explored remote rendering for interactive visualization on mobile devices.

Therefore, I implemented a remote-view dependant volumetric rendering web-service. Whereby the client transmits the projection and camera matrices to the server and the server returns a rendered image of the volume from that view perspective. This provides several advantages: the client has to send minimal data to the server, images can be highly compressed for fast transmission, the data is contained locally on the server, but most importantly the volume is rendered very quickly using GPU ray-marching.

4.2.3 HURRICANE SIMULATIONS

During our work on hurricane simulations I very quickly ran into the problem of scale as I started expanding the simulation space to additional states and historic hurricane paths. The computational time grew exponentially. Therefore, I needed additional techniques for scaling up.

I developed a distributed computing system to run our simulations en masse. The best way to distribute the work load was by running each simulation seperately as a single task, but to run the simulations all from the same path as a group. This way data transfer between slave-master nodes is minimial since the path parameters only have to be broadcasted once until the entire simulation group per path is finished.

We used

One particular issue I ran into was that in some cases simulations even from the same path would take variable time to complete. Since there is stochasticity within the simulation different runs of the same hurricane path will finish at different times. This can cause load imbalance on a distributed computing system. So I incorporated work stealing. If a node becomes idle because the simulations finish faster due to randomness it can steal work from other nodes. This also ties back into running all the simulations from one path as a group. Since the parameters for a particular path are broadcasted initially to all nodes no additional data needs to be transmitted. Thus, speeding up compution further.

```
**Result**: Produces density for a given voxel
$h_s$, $h_t$, $voxelSize$, $bbMin$;
$voxelCenter \leftarrow bbMin + (voxelSize * 0.5) + (voxelSize * globalId)$;
$density \leftarrow 0.0$;
$root \leftarrow octTree[0]$;
while $root.thread \neq -1$ do
    $nodeCenter \leftarrow (root.max + root.min) * 0.5$;
    $nodeHeight \leftarrow (root.max.z - root.min.z)$;
    $nodeWidth \leftarrow distance(root.max.xy, root.min.xy)$;
    $d_s \leftarrow distance(voxelCenter.xy, nodeCenter.xy)$;
    $d_t \leftarrow distance(voxelCenter.z, nodeCenter.z)$;
    if $d_s < (h_s + nodeWidth * 0.5)$ and $d_t < (h_t + nodeHeight * 0.5)$ then
        if $root.leaf$ then
            for $s \leftarrow root.cellRange.x$ to $root.cellRange.y$ do
                $cell \leftarrow cells[s]$;
                for $i \leftarrow 0$ to $cell.length$ do
                    $dataPoint \leftarrow data[cell.b[i]]$;
                    $density$ += $kernel(dataPoint, h_s, h_t)$;
                end
            end
            $root \leftarrow octTree[root.thread]$;
        else
            for $i \leftarrow 0$ to 7 do
                if $root.q[i] \neq -1$ then
                    $root \leftarrow octTree[root.q[i]]$;
                    break;
                end
            end
        end
    else
        $root \leftarrow octTree[root.thread]$;
    end
end
```

Algorithm 1: STKDE Pseudocode

Figure 27: Pseudo code for our STKDE algorithm.

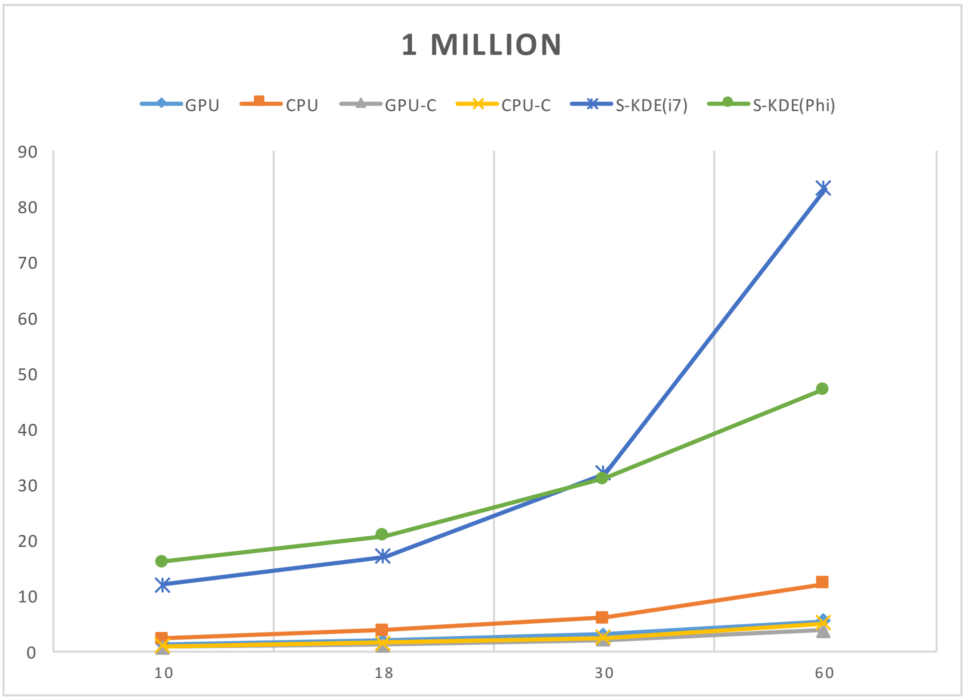


Figure 28: Performance results for 1 Million sample points using mvrnorms. S-KDE is the crop and chop KDE algorithm. GPU-C and CPU-C is our algorithm with coalescing.

CHAPTER 5: PLATFORM AND INTERACTION

In this section I detail the importance of platform and interaction as well as how the two are sometimes related. Platform is an important piece to situational awareness and visual analytics because it enables users to do significantly different things with knowledge acquired through visual analytics. Once I have developed strategies for analyzing and visualizing data at scale I then have to choose the best means for presenting and allowing users to extract meaning and usefulness through a platform and interaction. Interaction can be unique to the individual platform because mobile devices rely on touch based gestures. So complex multi-coordinate views that require a mouse must be redesigned and developed for use on a mobile device.

One specific type of interaction that I also focus on as our contribution is search by example: search-by-example is a set of direct manipulation techniques that enable users to gather meaningful results from complex data by presenting examples of what they want, obviating the need to construct complicated, often incomplete queries [16]. Search-by-example often involves some type of data analysis. In section 3 I cover several novel search-by-example techniques I have developed. While search by example for image based data is a well researched field I focus on the novelty of applying it specifically to LiDAR data. As well as evaluating a hypothetical hurricane path in real-time, in terms of relevant results from our ensemble simulation collection. This provides immediate, actionable results to emergency planners and first responders.

5.1 WEB

One of our major benefits with a web-based platform is accessiblity. It is plugin free and easy to use right from a web-browser. A web-based tool could be easily accessed and used with minimal hardware and software. Delmelle et al. [32] developed a web-based tool for visualizing Dengue Fever. Cho et al. [21] developed a web-based analytical system for visualizing temporal and spatial data in multi-coordinated views.

5.1.1 EVACUATION SIMULATION

As part of our work in evacuation simulations one of the challenges was bringing the tool to a web-based environment. This would allow stakeholders and other emergency planners to access the tool without installing complicated bulky software.

Our earlier system was difficult to provide to emergency planners and was limited in mobility. Therefore, I turned to web-based languages and tools to make it highly portable. All of 3D rendering is done using WebGL an implementation of OpenGL for web browsers. We use the PostgreSQL database with the PostGIS extensions to maintain all information related to the building geometry and for serverclient communication. The database is accessed running on the local machine with direct package calls. The reports section is an HTML/ JavaScript window inside a QT4 widget. This allows easy porting to mobile device browsers as well as easy dissemination to system users as seen in Figure 29.

All of the visualizations are directly supported on the web browser; the animation view is implemented as a WebGL application. WebGL is a browser-based implementation of OpenGL ES 2.0 and uses the GL Shading Language for shaders. Given that current implementations of WebGL can exploit graphics hardware, very little is lost in terms of geometry rendering efficiency, while application portability is greatly increased.

The WebGL application loads campus building data (via files in GIS native format); a triangulation is performed, followed by an extrusion to create a 3D model. These building models are then concatenated into a single buffer to improve rendering efficiency. The application follows the same procedure for campus footpath geometry. Animation data consisting of the occupancies centered around each area are loaded, followed by generation of 3D bars scaled and colored by the occupancy at each location. The building geo- metry is rendered using a lambertian lighting model with backface culling. Bars are drawn with a flat color, and no lighting to reduce shader computation. Screen space ambient occlusion is used to help distinguish the buildings and bar graphs from the background. The remaining views that include generation of congestion reports and exit statistics are implemented using web technologies (HTML).

All of the interactions involving specification of blockages and rerouting of occupants around inaccessible areas of the building are sent as requests to the server for processing and signaled back to the browser upon completion. This approach avoids significant data processing within the browser. The HTML5 functions related to the canvas for WebGL, websockets for persistent connectivity, and webworkers for true threaded browser operations permit this design and allow for a rich and portable evacuation application.

The simulation is manipulated by direct interaction over the 3D animation and report views. For instance, a blockage can be introduced via the 3D view, and simulation rerun to generate new (rerouted) paths for impacted evacuees; the report view is updated to reflect the situational change. Similarly, the interaction with the reports menus, charts, and so on temporally updates the 3D animation view. All such operations are performed in near real-time, as the computation is performed on simplified graphs.

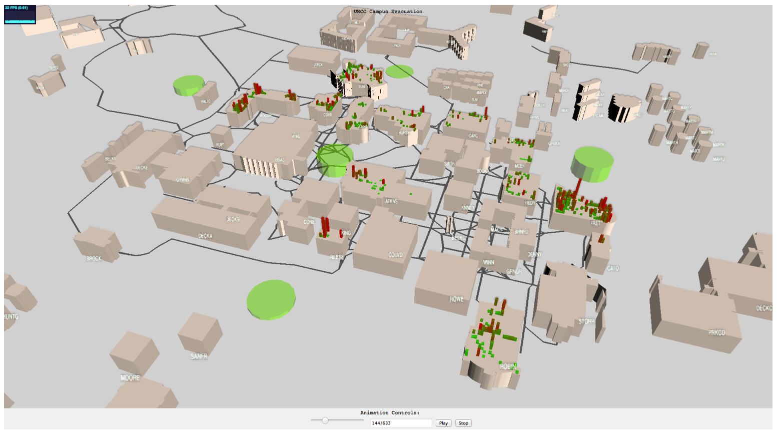


Figure 29: This figure shows our campus wide evacuation simulation in a 3D web interface using WebGL. The green cylinders indicate muster points where students, faculty and staff would gather. The height of the cylinder corresponds to the amount of people. Ontop of the buildings are colored bar graphs that show conges- tion in that area of the building. The color of the bar indicates the seriousness of congestion, where as the height indicates the number of people within that area.

5.1.2 STKDE FOR WEB

I followed the previous paradigm of using WebGL, an OpenGL ES version for web browsers, and establishing the STKDE computation and visualization as a webservice [40]. As I detail in Chapter 4, I developed this approach with the goal of incorporating it into a high level language like Python. I calculate the STKDE in a Python framework (permitting easy integration with existing frameworks and data analytics tools) and embedded it into a webservice on a GPU server.

In order to visualize the STKDE, I needed a 3D mapping system that could be integrated with existing web-based tools and multi-coordinated views. All of the existing mapping tools like Google Maps, Apple Maps, Bing, etc. did not have extensive 3D capabilities that suited our needs. The mapping tool also needed to have an easy way to incorporate custom 3D geometry. In prior work I developed such a mapping system to use on mobile devices using OpenGL. Our mapping system was built in 3D, but used a 2D projection. Therefore, little work had to be done to equip the system with 3D capabilities by porting OpenGL over to WebGL using ThreeJS.

As part of the STKDE I implemented several interactions. The first is rotating. I utilize an orbiting camera fixed around a focal point. A user can rotate the camera around the focal point by dragging with the left mouse. This is typical interaction you would find in a 3D application. The second is panning, using the right mouse a user can pan the 3D view based on the viewing angle with relation to the up vector of the 3D scene. The third navigation interaction is zooming, which can be done using the mouse wheel. Lastly, there is interaction with the 3D structures produced by the space-time kernel density estimation. These interactions include slicing. The view is also linked with the other existing views, both the 2D spatial view, and the temporal view.

Klemm et al. [78] incorporated substantial usage of slicing planes in their work on 3D regression heatmap analysis. For the purposes of our work I incorporate two different types of slicing. Temporal slicing involves selecting a region in time and viewing just that slice in a 2D spatial view. Spatial slicing involves selecting a region in space and viewing the slice across all time just for that region.

I provide a time graph that depicts the entire time-signature across the entire data set. A user brushes along this global time view to make a selection. Then a zoomed in portion of the time graph for that selection is created to show a more detailed signature of temporal events. The time graph selection generates a corresponding bounding box in the 3D view that specifics the region in time that was selected. In the 2D view I generate a heatmap using just a kernel density estimation.

As detailed in 3.1.5, spatial slicing is important because the STKDE volume is often not symmetric around its perpendicular axis. It may be stretched in one direction or have a bump; it may merge with another volume in a particular direction. The temporal behavior of a slice in one direction can thus be significantly different than that in another direction, and this difference can be meaningful, as I will discuss further in the use cases below. Therefore it is important to quickly and easily set up slicing so that volumetric features can be accessed with reasonable precision and lack of ambiguity. In the same way that temporal slicing shows a distinct region across time, spatial slicing shows a slice in space. I looked at several options for implementing the interaction for spatial slicing. Doing the slicing from the 2D view would be possible and could be done by using axis-aligned slicing planes. However, 3D interaction can be complex and hard to manage. Our goal is to make the tool as straightforward as possible, but there could be situations where the slicing plane needed to be angled away from an axis. Also additional visualizations would have to be included in the 2D view in order to associate with the 3D structures. An example of this might be an outline of the 3D structure overlaid on the 2D view to help with coordination. All of this posed a unique challenge and ultimately I opted for doing the slicing in the 3D view. Therefore, I had to create easy to use and intuitive 3D interactions. In the 3D view I already make use of the left mouse and right mouse for panning and rotating. Instead of adding elaborate menu items for changing interaction modes, I used double clicking in the 3D view to select regions on the wireframe bounding box of the space-time kernel density structures. The processes works as follows. A user selects a point on the wireframe bounding box. This generates a perpendicular dotted line from the wireframe selected. At the end is a large point that the user can double click to complete the line segment and create the slicing plane. I incorporated this option because it would be difficult for a user to make a perfectly axis-aligned slicing plane if he or she wanted to. Otherwise, the user can select any other point on the top of the wireframe to create an angled slicing plane. This is the most minimal amount of effort and bypasses the need for other more complex interactions. The contours provide a detailed analysis. The user can see how they spread in the selected spatial direction over time. In addition, the image shows temporal and spatial components aggregated at the right side and stop, respectively. Thus, for example, the right side shows the timeline along the selected spatial direction.

5.2 MOBILE

Mobile devices today play a unique role in interactive visual analytics. They allow for decision making on the go or in the field. This is an absolutely crucial area for developing novel interactive techniques and visualizations. I expanded the field by developing new work for interactive visual analytics focused on understanding complex space-time data using the STKDE. I evaluated our approach using hurricane simulation data and through a user-evaluation.

Interactive visual analytics has expanded in recent years on mobile devices, but it is still an active area of research. I cover prior work in tablet-based interactive visual analytics as a precursor to the new work I later discuss. I also review work in 3D interaction on tablet devices because the STKDE involves 3D interaction [40]. Chittaro et al. [20] outlined the significant differences of doing information visualization on mobile devices. I also followed the direction of Games et al. [51] for effective data visualization on tablet devices and Blumenstein et al. [8] detailed how to evaluate information visualizations on mobile devices. Lee et al. [86] documented design principals for visualization interaction beyond using a mouse and keyboard. Wigdor et al. [148] documented design methodologies for user interfaces for touch-based gestures. Craig et al. [27] documented design guidelines for mobile visualization and interactions based on device limitations. They detailed that implementation is key for interaction otherwise it is easy to overwhelm the user.

Escobar et al. [47] detailed information visualization design on the Adkintun mobile app. Rzeszotarski et al. [118] created Kinetica for multi-touch data visualization. In 3D Modeling by the Masses [41], the authors created a mobile interactive visual analytics application for mapping indoor buildings. It utilized split-coordinated views for 2D and 3D navigation, transparent overlays, minimal and hidden menus. Sadana et al. [119] detailed interactive scatter plot visualizations for tablet devices. I looked at this work as reference for visualizing and interacting with the results of our spatial slicing interaction for the STKDE. Drucker et al. [38] developed TouchViz by comparing two different interfaces and evaluating the performance between gestures and control panel interactions. Sadana et al. [121] examined advanced gesture interactions for visual analysis on tablet devices. Au et al. [4] evaluated multi-touch gestures for 3D object transformation. Lopez et al. [91] explored touch based navigation of 3D objects on a mobile device that also connected to stereoscopic displays.

In the first section we detail our crowd sourcing mobile app. Then we discuss our hurricane ensemble visualization mobile app. Lastly, we detail the use of the STKDE on a tablet device.

5.2.1 CROWD SOURCE

The mobile system also had to support numerous complex interactions in-order to create building models. The mobile interface also had to be simple and easy to use. Creating 3D geometry on a mobile phone with limited screen space and touch gestures is challenging. Therefore, I limited the feature set to a few basic objects. As well I restricted the modeling environment to a grid based system in which users selected tiles on which to create geometry. Using the tile system also means it was easier to maintain continuity between different 3D models. The user was presented with two modes first a creation mode in which tiles could be selected and objects created. Then a modify mode in which users could select objects

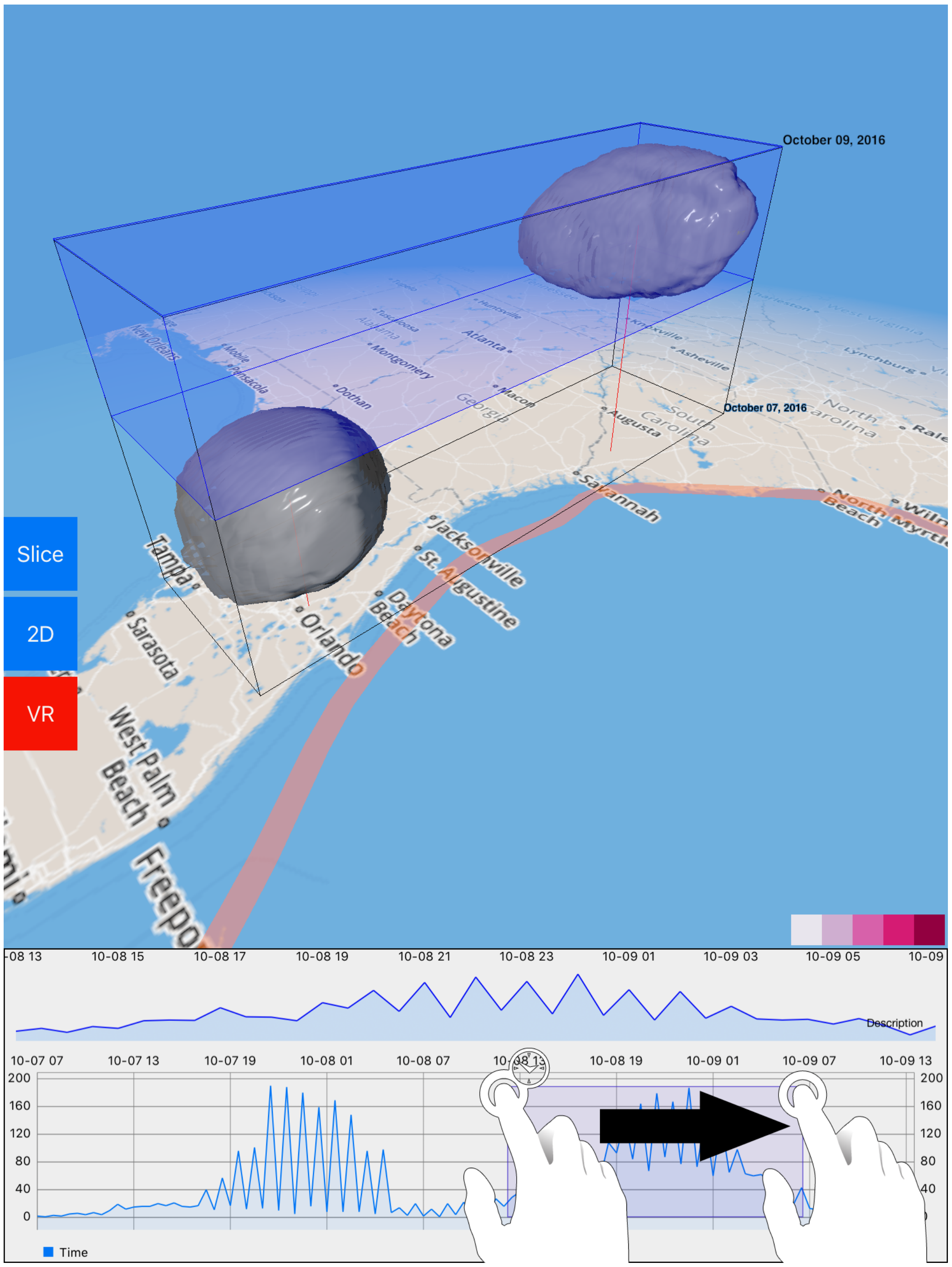


Figure 30: An example of the time graph with zoom + brushing. The X axis is time and the Y axis is the number of outages that occurred at that given time. The gesture used is a long press gesture plus drag in order to make temporal selections.

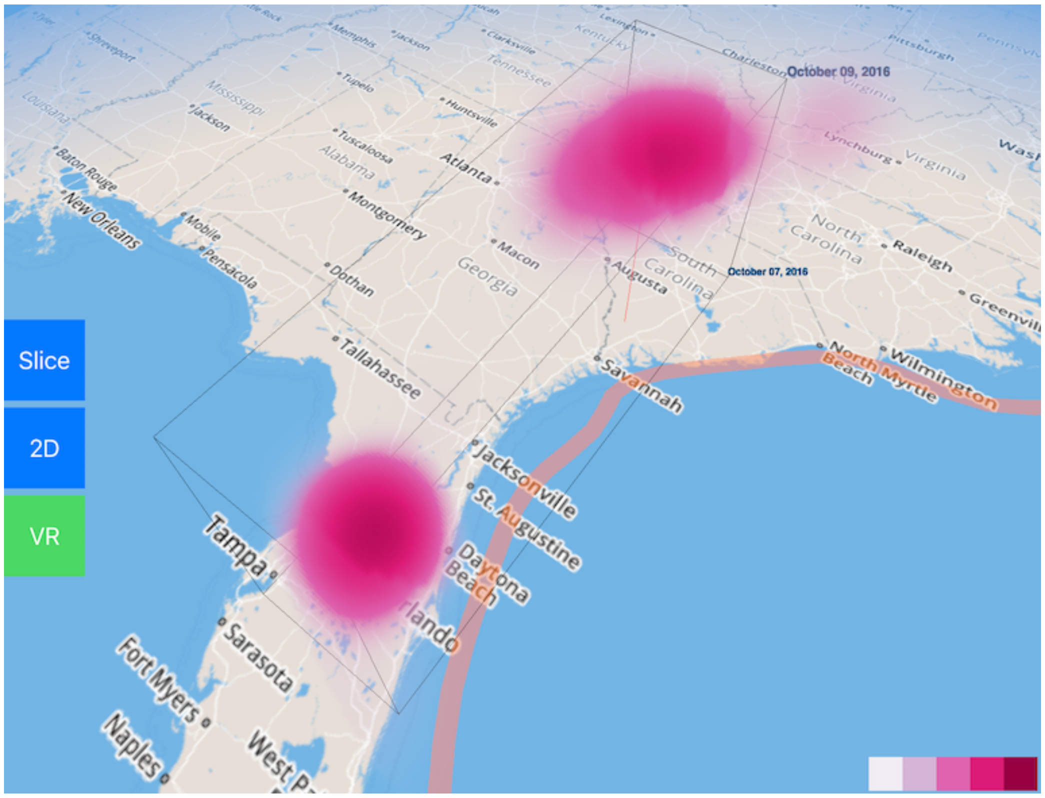


Figure 31: An example of space-time volume containing two STKDE volumetric structures.

to modify them like rotate or move them on the grid. Users could also group objects to build hierarchies. They could also copy items and perform deep copies duplicating large hierarchies of objects to minimize the effort needed to create geometry. Lastly there was a voting and ranking system. Users would randomly receive a model from the server that had the fewest amount of votes. A user could then vote on the model ranking it on two different metrics similarity and completeness. The two scores were then aggregated together to compile a total combined score for the model.

5.2.2 HURRICANE SIMULATIONS

In prior work on hurricane simulations [43] I focused on two interaction techniques, filtering and probing. Filtering is a key function that allows for the removal of unwanted features or events. In such a large simulation space, it is crucial for a user to be able to focus on specific commodities and hurricane paths. More particularly the combination of the two together allows for complex filters. This helps users navigate the simulation space by visualizing different parts of the simulation in a very easy manor. The first type of filter is a commodity-based filter. Each commodity is separated into specific categories that share a common area. For example all the infrastructure features like ports, train stations, airports are separated into a transportation category. By having high-level categories, users can more easily pick what areas they would like to filter instead of navigating through lots of very specific features. Grouping similar features into categories also allows for easier comparison. I allow users to filter specifically on commodity types like transportation and electricity. This allows for quick comparison between different commodities.

The second type of filtering is hurricane path based. Our model has numerous hurricane paths that create a probabilistic outcome. Users can select specific hurricane paths to see the probabilistic effect of each impact. But, the real core of this feature is the filtering of multiple hurricane paths. By filtering on multiple hurricane paths pertinent to a region a user can get encompassing results for that region or quite specific results (e.g., what hurricane path is most likely to do damage to this particular infrastructure feature in my local emergency response area, and when).

Probing involves interactions that allow users to visualize a selected region in a free-form manner [13, 12]. A user can directly draw onto the map with his or her finger, drawing a bounding area around a region. This is an extremely important analysis tool because it allows the user to drive the analysis and focus on what is important to their geographic area. Anything within the bounding region will then be analyzed and returned to the user in a separate window. When selected, a separate window will appear that initially shows the time distribution for all the features within the region. A user can quickly see any peaks in time in this window. Secondly, the user can then view the breakdown of what feature classes are in the selection and how many were disabled. This allows users to see what features were disabled, how many were disabled and the time distribution of those features, permitting a very fast analysis of regions of interest. It also allows users to pinpoint features that have a high probability of being disabled. Most importantly it allows a user to compare multiple regions for joint analysis.

Each probe selected by the user is copied and saved for additional viewing. This way users can select multiple probes across many different areas and filters. Users can probe different time segments as well as different hurricane paths and commodities. Along with the selected features a screen shot of the region is attached to the data to give geospatial context to the region that was selected. All this information is presented in a secondary window where each probe is represented as a card with a map image of the selection and a time distribution of the events from the probe. When the user selects a specific card the view transitions to a detailed breakdown of that specific probe. In this way users can return to further analyze individual events and time distributions from prior selections they have made.

5.2.2.1 PROBE SELECTION

Probing involves interactions that allow users to visualize a selected region in a free-form manner. As shown in Figure 32, a user can directly draw onto the map with his or her finger, drawing a bounding area around a region. This is an extremely important analysis tool because it allows the user to drive the analysis and focus on what is important to their geographic area. Anything within the bounding region will then be analyzed and returned to the user in a separate window.

5.2.2.2 FILTERING

In addition, filtering is a key function that allows for the removal of unwanted features or events, as shown in Figure 8. In such a large simulation space, it is crucial for a user to be able to focus on specific commodities and hurricane paths. More particularly the combination of the two together allows for complex filters.

This helps users navigate the simulation space by visualizing different parts of the simulation in a very easy manor. The first type of filter is a categorical-based filter (Transportation, Electric, Oil and Gas). Each feature is separated into specific categories that share a common area. For example all the infrastructure features like ports, train stations, airports are separated into a transportation category. By having high-level categories, users can more easily pick what areas they would like to filter instead of navigating through lots of very specific features. Grouping similar features into categories also allows for easier comparison. I allow users to filter specifically on category types like transportation and electricity. This

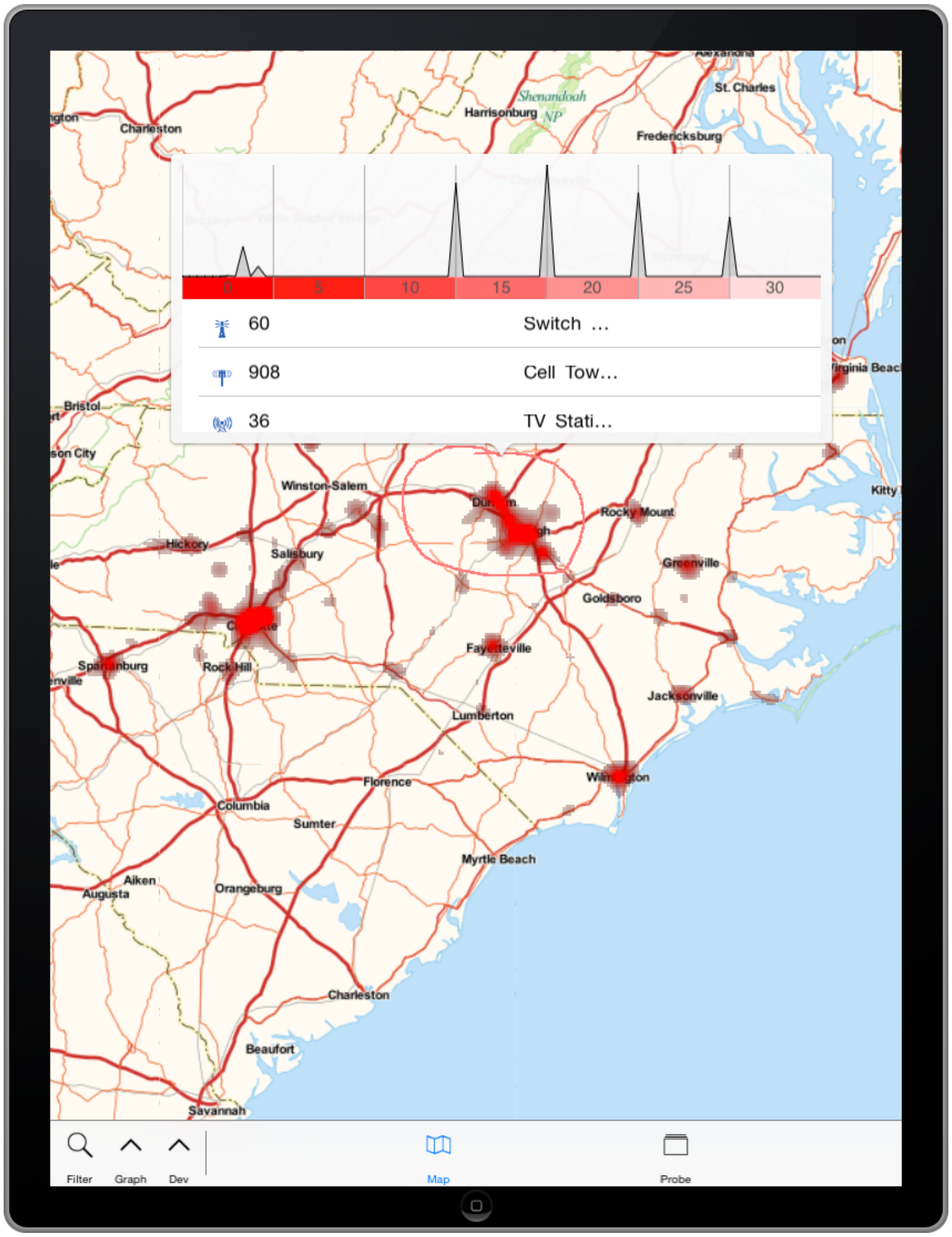


Figure 32: Ensemble Simulation Probing Interaction.

allows for quick comparison between different categories. The second type of filtering is hurricane path based. Our model has numerous hurricane paths that create a probabilistic outcome. Users can select specific hurricane paths to see the probabilistic effect of each impact. But, the real core of this feature is the filtering of multiple hurricane paths. By filtering on multiple hurricane paths pertinent to a region a user can get encompassing results for that region or quite specific results (e.g., what hurricane path is most likely to do damage to this particular infrastructure feature in my local emergency response area, and when).

5.2.3 STKDE FOR MOBILE

Multiple-coordinated views are a powerful visual analytics tool [115]. But applying multiple-coordinated views to tablet devices is not a straightforward problem. Integrating the STKDE onto a tablet-based device requires multiple coordinated views. I followed Sadana et al. [120] who evaluated design principals for multiple coordinated visualizations on tablet devices. In this section I discuss how I designed the interface and interaction specifically for the STKDE on a tablet device.

5.2.3.1 3D + 2D VIEW

Due to nature of tablet devices and screen size, I looked at alternatives for setting up and visualizing these multi-coordinated views.

I tried displaying both at the same time but the screen size of the tablet made this impractical. Therefore, I opted for a simple toggle menu button between the 2D and 3D views. The views are still coordinated in that panning and zooming in one will be reflected in the other, but rotating is only done in the 3D view.

The 3D view supports panning, zooming, and rotating using known multi-touch 3D user

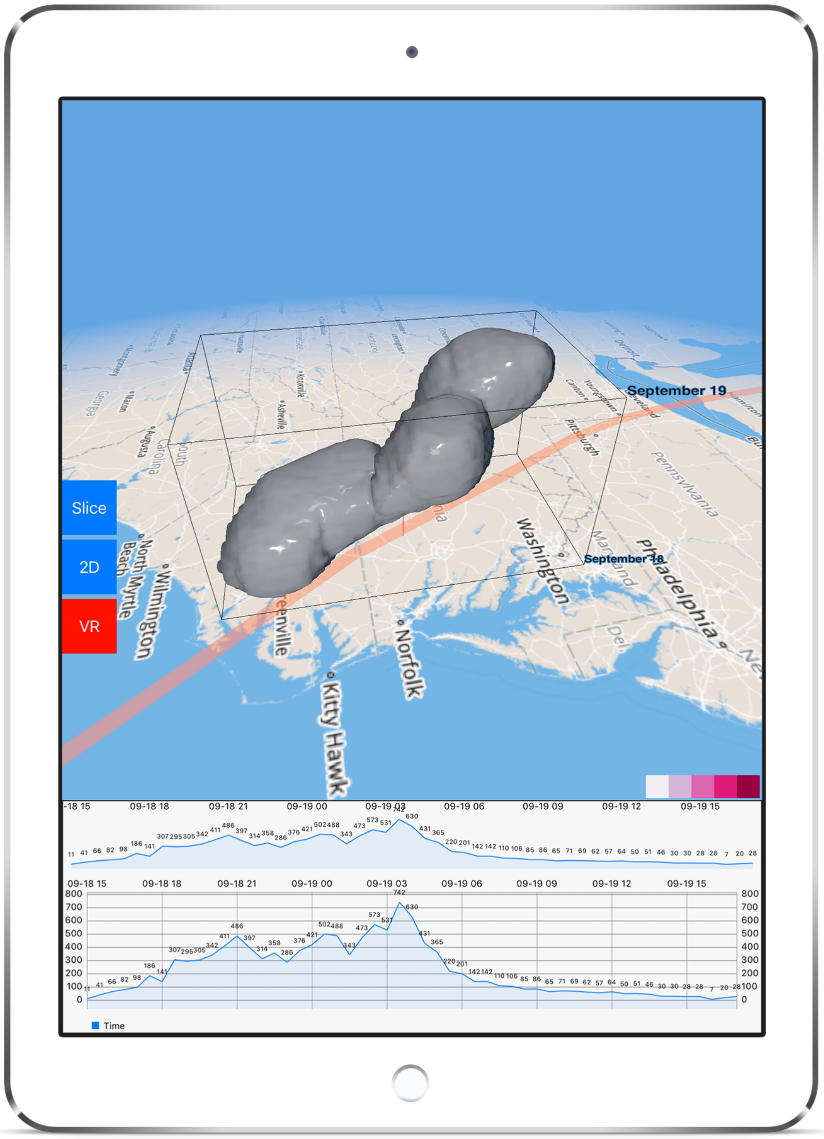


Figure 33: An example of our entire visual analytics (VA) interface depicting the 3D view.

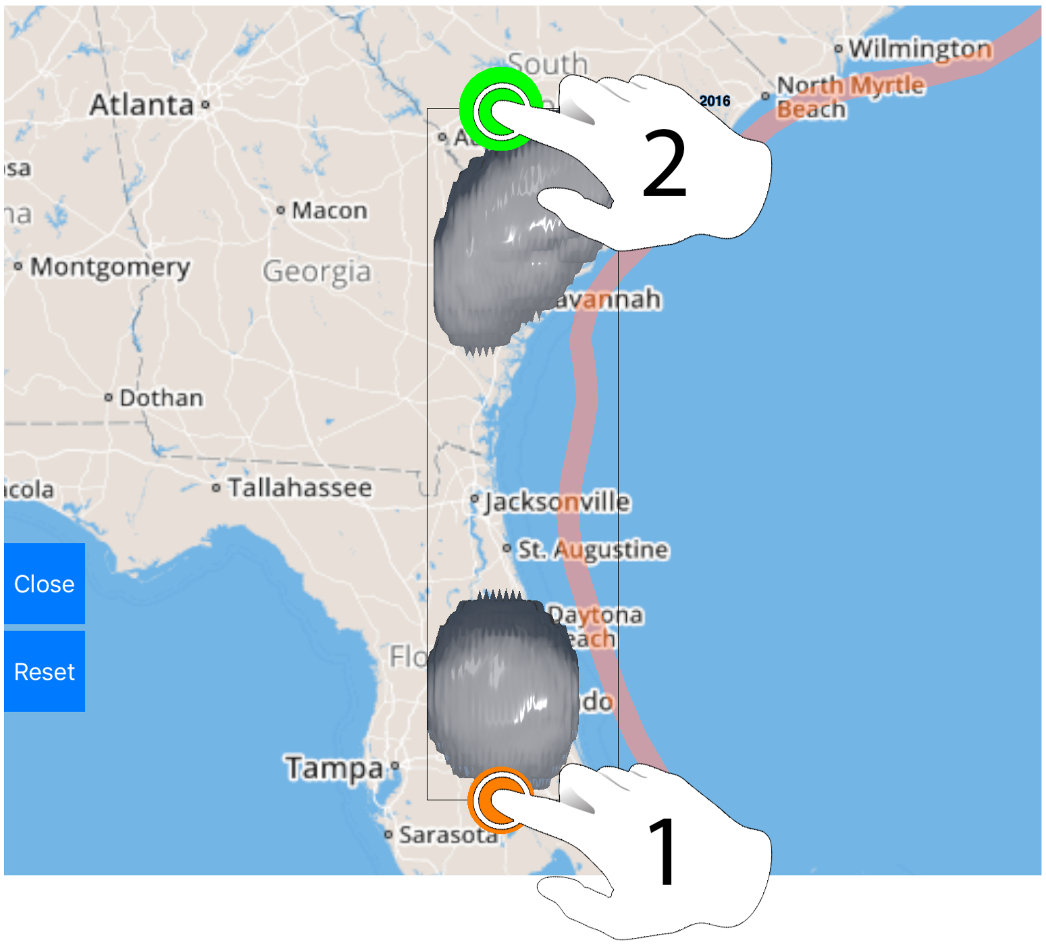


Figure 34: An example of the spatial slicing viewport, whereby a slicing point has been selected and a corresponding green point appears to create a perpendicular slicing plane. The first point is created by double tapping by the hand (label 1). The spatial slicing plane can then be completed by double tapping on any adjacent edge or on the green point to create a perpendicular plane with the hand (label 2).

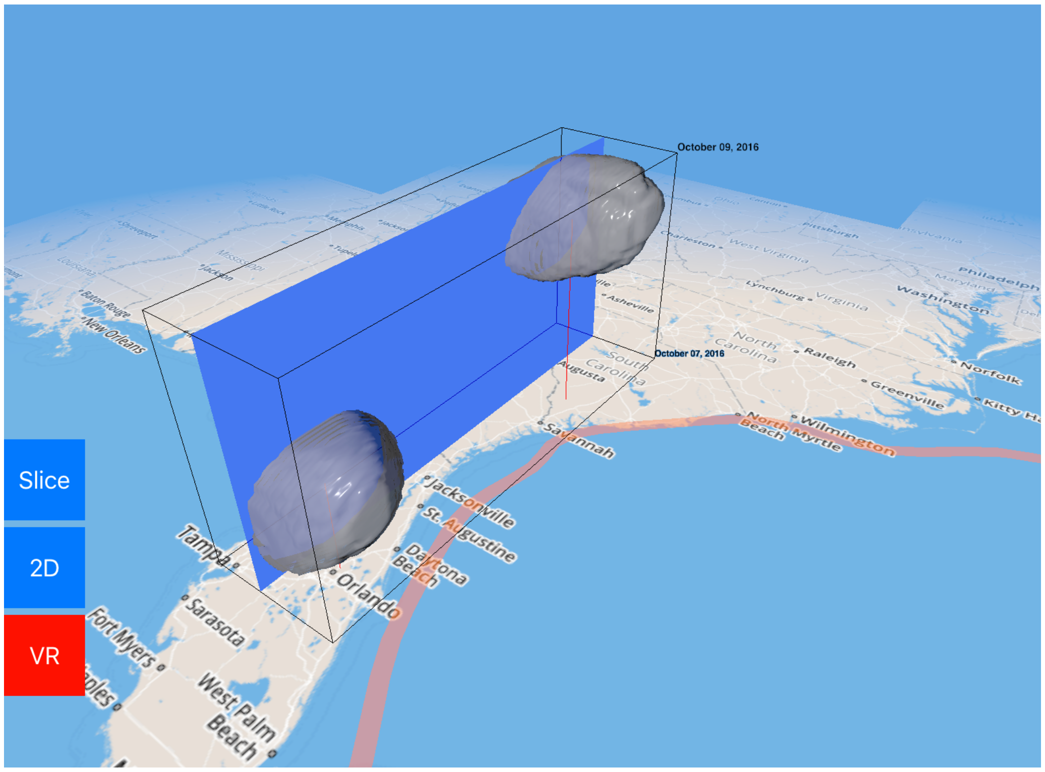
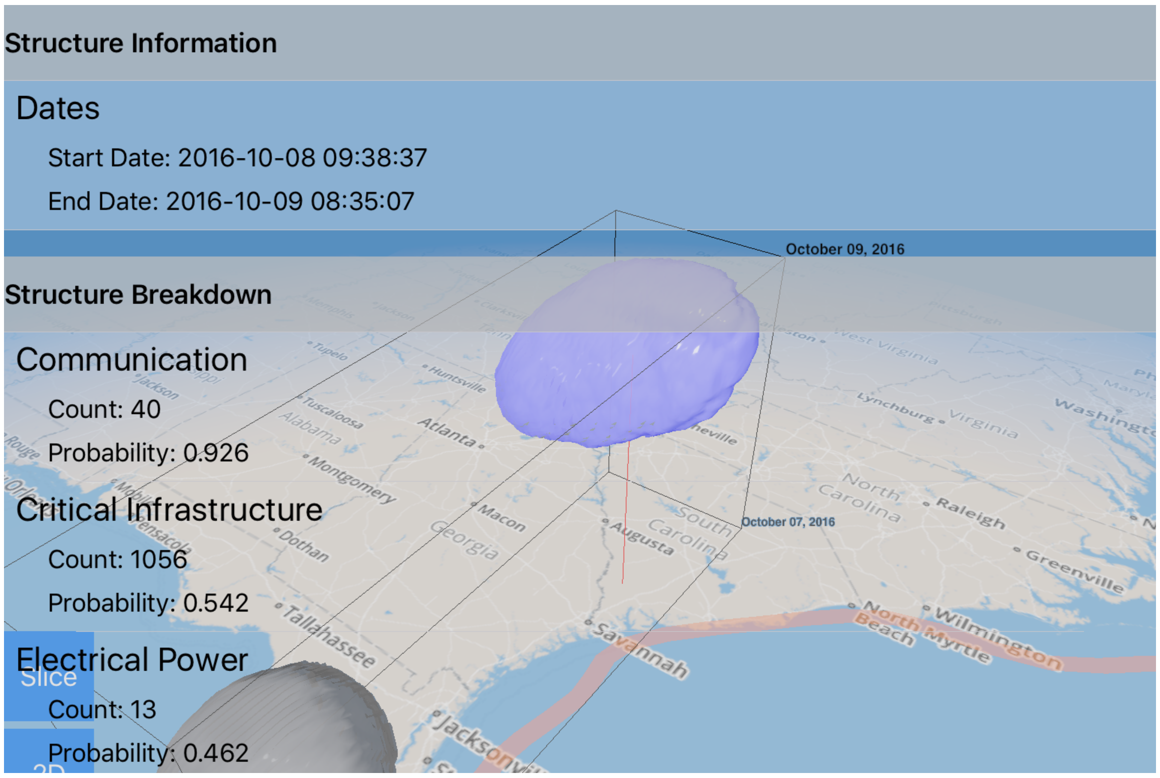


Figure 35: The result after completion of the spatial slicing plane in Figure 34.

interface techniques. For panning, I use a single finger drag gesture. Panning is dependent on the orientation of the 3D camera. It works by ray casting from the drag gesture location and finding the intersection with the geospatial map. I use this to generate a directional vector to pan the 3D camera. Zooming is performed through a pinch gesture. Lastly, rotating is performed by a two finger drag gesture, where one finger is left stationary and the other performs the rotation. The rotation is done by orbiting around the focal point of the 3D camera, whereas dragging along the X axis of the view pivots the camera up and down. Dragging along the Y axis of the view orbits the camera left and right. To simplify rotation movements, I also lock the rotation on the Y axis to 45 degrees between the map surface and the camera.



Structure Information

Dates

Start Date: 2016-10-08 09:38:37

End Date: 2016-10-09 08:35:07

Structure Breakdown

Communication

Count: 40

Probability: 0.926

Critical Infrastructure

Count: 1056

Probability: 0.542

Electrical Power

Count: 13

Probability: 0.462

Figure 36: This figure shows an example of an individual STKDE structure selected which is highlighted in blue. As well as a transparent overlay detailing the overall breakdown of features within the selected structure.

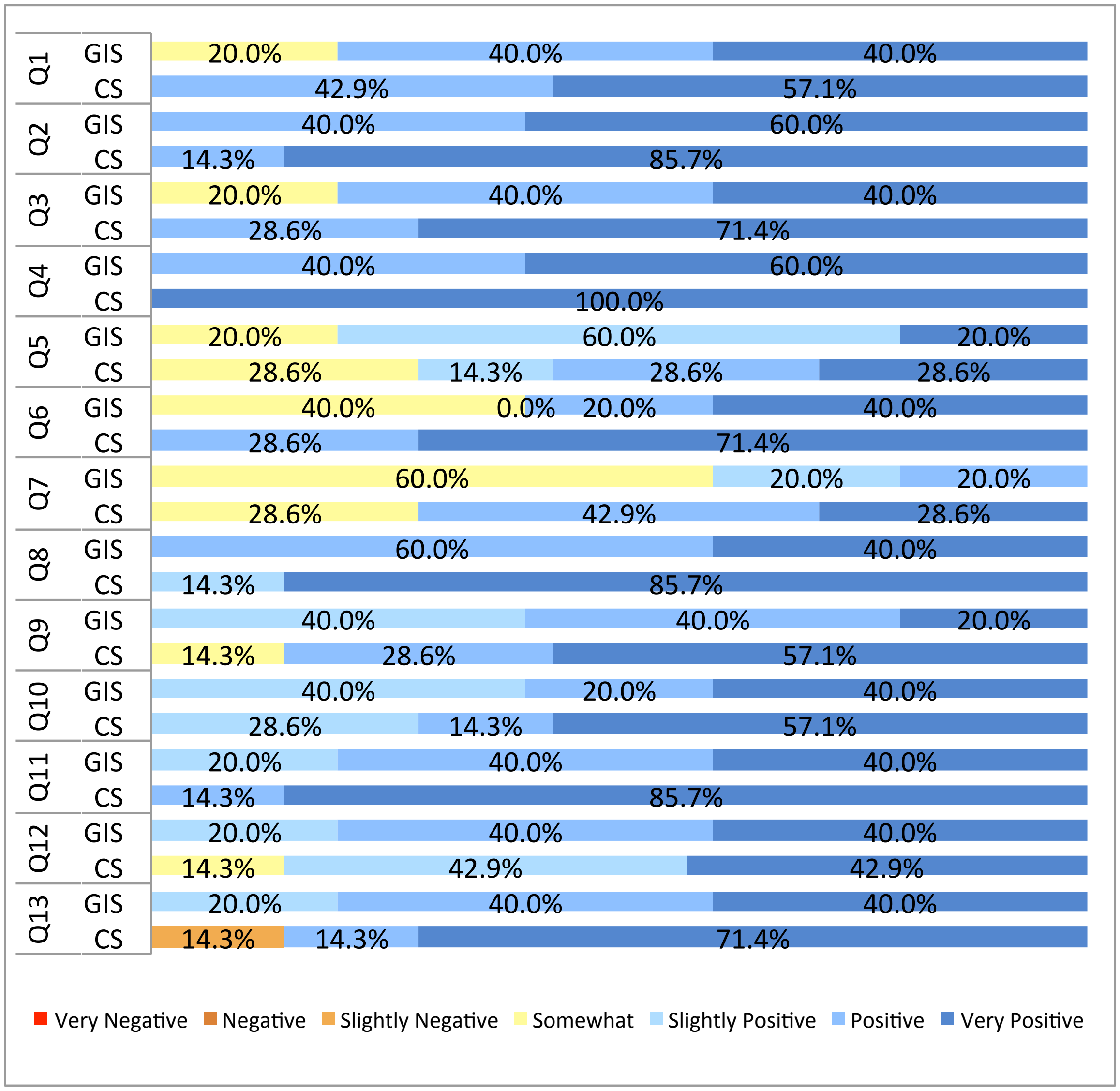


Figure 37: The combined results of the responses from each participant group. Each question is broken down as a total percentage of how many participants in that specific group answered very positive, very negative or in between.

5.2.3.2 TEMPORAL VIEW

The temporal view is the second most important view in our visual analytics tool. Therefore, I allocated approximately twenty-five percent of the screen space for the temporal graph. The temporal view depicts the aggregated time-line of all events that occurred across the entire data-set. In our case this was the outages of specific features like hospitals, police stations, etc. as seen in Figure 30. The temporal graph is always visible. I use a brush + zoom time graph. In order to do brushing, I use a long press + pan gesture (depicted in Figure 30). This way the user has to enable the drag selection first. I decided to use this implementation because the brushing action is linked to other views and web-services. I did not want accidental touches triggering these connected actions, so I made the brushing a distinct, deliberate action. I also provide visual feedback on the completion of the long press with a radiating circle that fades out. This lets users know they can begin brushing on the time graph.

5.2.3.3 TEMPORAL SLICING

In this section I detail how I incorporated temporal slicing. Temporal slicing is an important interaction because it allows the user to make a selection for a particular time or time range that can then be viewed in the 2D view, but also in the 3D view. This gives spatial context, but it also gives insight into the relationship between how events occurred over space-time. In order to perform temporal slicing I use the brush interaction on the temporal graph. Once the user performs a selection through the long press + drag gesture (Figure 30), I compute a KDE within the 2D view based on the selection as shown in the bottom image in Figure 5. The KDE is computed on the server and sent back to the client as a web-service. Secondly, I also update the 3D view with a blue bounding box that depicts the selected temporal region over the STKDE. This gives spatial and temporal reference of the time graph selection to the STKDE structures as shown in the top image in Figure 5.

5.2.3.4 SPATIAL SLICING

In this section I detail how I incorporated spatial slicing (i.e., with a slicing plane perpendicular to the spatial plane). Eaglin et al. [40] chose to do spatial slicing in a 3D view whereby a user selects points on the bounding box of the STKDE volume. This technique worked well with a mouse due to the precision of clicking. But for the tablet, this was not a feasible option due to the dependence on touch and gesture interaction. Instead I chose to simplify the process by turning it into a 2D problem.

From the 3D view a user selects the spatial slicing menu option. Selecting this option then transitions the 3D viewport into a 2D parallel projection. The transition is animated in order to provide context for the 2D view in the 3D view. The 2D parallel projection is oriented with respect to the geospatial bounds of the STKDE volume. Once the 2D view has finished animating, the user can then double tap anywhere along the bounding box to generate a slicing point. Since touch is not as precise as using a mouse, I select the closest point on the bounding box from the touch point. Once the user generates the first slicing point a corresponding large green point appears. This point allows the user to create a perpendicular slicing plane by tapping on the green point. If the user wants to create an angled slicing plane, he or she can select an adjacent location to create an angled plane. The selection process is illustrated in Figure 34 with the final slicing plane shown in Figure 35.

Once the slicing plane is created, results are generated server-side and sent to the client. An interactive scatter plot is then generated from the results as seen in Figure 7. The X value of the scatter plot is determined by computing the spatial distance in miles of the specific feature i.e. hospital from the start point of the slicing plane. The Y axis is the elapsed time in minutes with respect to the first feature disabled. Thus the scatter plot shows the times when and locations where (in this case) outage events pile up. Figure 7 shows clearly the details of behavior within and between the two volumes of outages, including trends in time. Note that the two STKDE volumes are essentially separated in time with small temporal overlap. The scatter plot results are binned spatially and temporally into a vertical and horizontal bar chart. This allows for easy and quick interaction to evaluate the details in space-time. The scatter plot is also fully interactable and allows brushing and selection.

5.2.3.5 STRUCTURE SELECTION

In many cases unique and separate structures are generated by the STKDE. There is a need to be able to select specific structures and analyze those in particular. Therefore, as part of this novel work I spent a significant amount of effort creating a solution that worked and was fast enough to incorporate with the STKDE calculation.

To make selections a user simply performs a double tap gesture in the 3D view over the desired structure. Then using ray casting, the specific structure is highlighted in blue and selected as shown in Figure 36. Since the underlying data is too large to send to the tablet device, I run an additional web-service whereby I aggregate and extract the corresponding data for the selected structure and send it back to the client. The data is displayed in a transparent overlay. I show the start and end dates of the structure and the structure breakdown overview. The breakdown overview details the outages organized in groups by type, such as communications or electrical power. It also shows the overall count and the average probability of outages within each group. An example of this is shown in Figure 36.

5.3 SEARCH BY EXAMPLE

Search by example is a powerful technique that allows users to provide an example of what they are looking for as a query. But to implement such techniques can often require several complex steps. First how is the interaction performed to generate the query or example. Secondly, how is the example then used to generate new analysis or visualizations. I have created two search by example techniques for our LIDAR analysis work and our Hurricane simulation analysis.

5.3.1 REMOTE SENSING

In this section I detail the LiDAR search-by-example technique. In 2.3 I discussed the overall LiDAR data and analysis methods. The analytical process described in that chapter is the backbone of how I generate new examples.

The entire goal of how I encoded attributes into images was for this step of thresholding. The process of encoding values into the RGB spectrum had to be meticulous because thresholding is very sensitive. A table showing the analysis to RGB conversion can be seen in table 1. The next step after thresholding is boundary analysis and blob extraction I also perform this on the GPU in order to achieve the up-most speed in searching. I perform blob extraction through marching squares. The marching squares algorithm is embarrassingly parallel and fits well into our GPU workflow. Lastly I calculate an image histogram for each blob to further analyze later. Each histogram for each blob can also be calculated on the GPU depending on the size of the blob. I combine all of these steps into a continuous process whereby I keep as much information as possible on the GPU to minimize data transfer from host to device and vice versa. This allows us to very quickly find results from searches on very large datasets in realtime. Using a similar hardware setup for the encoding process and 1024 x 1024 image sizes I are able to search approximately 52 processed LiDAR patches per second.

Once a user makes a selection I isolate that part of the image and run modified k-means clustering on the selection to extract the four most dominate colors as well as the radius of the clusters that I use to control the range of thresholding. Due to its nature k-means does not produce the same results each time by choosing random seeds. This is significant because thresholding is very sensitive to small changes; different clusters from the k-means could result in quite different results.

Then I run boundary analysis and blob extraction. As previously mentioned I perform these operations on the GPU for scale as well using marching squares. Each blob is then further analyzed and compared to the original selection through a histogram comparison and a bounds comparison.

5.3.2 HURRICANE PATH DRAWING

As I discussed in prior chapters with regards to our hurricane simulations, one of the major problems was evaluating hypothetical hurricane paths in real-time. Since the ensemble simulations can take hours to days to run, it is not feasible to do so via a web-service. Therefore, I have developed several techniques for generating results that can be computed in reasonable time.

In order to reduce the computational complexity significantly I turned to machine learning. The goal of this search-by-example technique is to allow a user to define a hurricane path and then extrapolate a hypothetical outcome given the ensemble simulation results. The path itself is essentially the query.

To generate a hypothetical path a user performs a press + drag within $HurVA$ to generate a path. This path is then sent to the server where it is analyzed and a subset of the ensemble simulations is extracted based on the drawn path.

CHAPTER 6: EVALUATION

In this chapter I detail the evaluation portion of the work I have done. This includes use-case studies, user studies and usability tests. Lastly, I detail final work in evaluating our hurricane simulations on critical infrastructure and taking it further.

6.1 EVACSIM AND EERC

Our goal of aiding emergency responders and planners began with the work in evacuation simulations and mobile response to provide better situational awareness [54]. Evacuating large buildings or stadiums is an important need for planners, but at the time of this work most current solutions were impractical for real-world situations, or handling scenarios with changing elements. To support this work several tools and systems had to be developed. Complex building models and graph techniques had to be developed to creating a real-time application. Multiple visualizations and reporting tools had to be developed in-order to provide accurate and important information to planners. Interactions had to be developed to allow new situations to be evaluated and easily explored.

The goal of the mobile response system was to give first responder in an emergency scenario better situational awareness inside of large buildings. The emergency mobile response system consisted of a client-server system with a command center and numerous mobile devices all communicating together. The first responder using the mobile device could update his or her position using the mobile device inside of a building. This updated information would then be visible on the command center system which would give a commander more situational awareness. From the command center commands, routing

Figure 38: Campus emergency response exercise carried out using the mobile emergency response system with campus police. The exercise was conducted in a multi-floor campus building to find a potential shooter.

information, and potential issues like a shooter or victim could be sent to a specific mobile device. These commands were sent via a visual notification as well as vibrating the device to alert responders. The mobile device contained complete building layouts and allowed for easy navigation between floors. It also allowed for 3D visualization of the building so first responders could visually see their routes between floors.

As part of this work several exercises were run with campus police cooperation to evaluate the systems and tools. The mobile response system was used in three separate exercises where simulated emergencies like a potential shooter inside of a campus library were carried out. The campus police used the mobile response system to find, locate, and update their position inside of a large multistory building on campus. The police found the system to be easy to use and integrated quite well into their existing protocols. We collected this information through an informal discussion with them directly after the exercise.

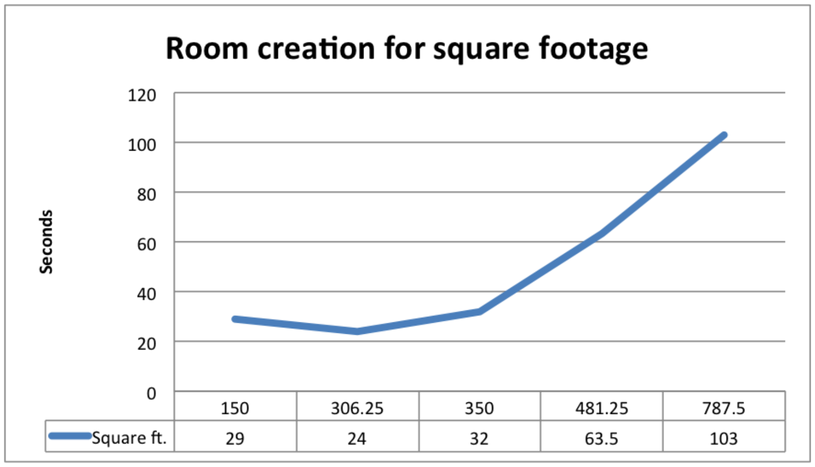
Secondly, the evacuation simulation was carried out in part with the campus police chief in several hypothetical scenarios in which the police chief had to make in the moment decisions based on the results from the evacuation simulation. The police chief found the system to be very easy to understand and use. Zhou et al. [156] recently explored new ways of visualizing indoor locations for emergency planning. Such techniques could be further adapted to our work in order to provide better visualization tools and disseminate knowledge to planners and first responders. One issue we discovered was the possibility of dead-zones or areas with no wifi or cellular signal inside of campus buildings. These areas are usually in stairwells or elevators in which police would often stop before proceeding. This presented a unique problem. Officers could not use their devices at these crucial times like receiving updates or updating their own position. They had to wait until service was restored.

During exercises with campus police an important need for having more mobile systems and applications became apparent. The campus police chief used an iPad tablet as part of his own protocol for managing communications with outside participants like local law enforcement, and fire department. He stated that being able to use everything from his iPad would be extremely beneficial as the campus police operates from a remote command center in emergency situations. This also presents a new opportunity for mobile applications to address emergency response situations and planning that can be studied in future work.

6.2 REMOTE SENSING

In this section I detail our work, 3D Modeling By the Masses [41]. As part of the work in evacuation simulations and mobile response the most critical issue was collecting, organizing, and analyzing building data to construct 3D models and indoor routing paths.

A major issue was that it relied heavily on the underlying generated data for building layouts and structures. Acquiring this data and generating the necessary parts required. It often required manual refinements and existing knowledge of building layouts to fix errors.



	150	306.25	350	481.25	787.5
Square ft.	29	24	32	63.5	103

Figure 39: This figure shows the creation time using our tool per square ft.

As well most existing data is in CAD format that can be hard to manage, out of date, difficult to convert into a 3D model and generate a network graph for routing.This problem presents an opportunity for future work in developing new ways to generate 3D building data that can further be used and more easily integrated into these applications.

Using existing CAD drawings and data are often hard to manage, out of date, and contain extraneous information. These files are hard to process automatically and require manual refinement and fixes to verify the accuracy. Existing systems like wikipedia rely on user submitted and validated information to provide fairly accurate and refined data. Therefore our aim was to create a tool for crowdsourcing the mapping of buildings to acquire better data for the use in our existing Evacuation and Mobile response tools. In order to verify the validity of the data the system utilizes a voting and approval system. Therefore, the crowd not only generates the data but also evaluates it. Since this data is used in critical applications like evacuation and emergency response, data validation is extremely important.

Users were able to contribute to repositories of specific models. For example one repository was created for the first floor of our computer science building. The repository model was referred to as the public in which users could download onto their mobile device and make changes. Once a user made a change they could submit that modified addition to the server for voting. Users could then track their submissions and see how well they are doing. On the server side databases had to be created and structured in a way that could easily handle queries for different models and different parts of the model hierarchy. The geometric models could also get quite large so on the fly zlib compression was added to minimize network traffic to mobile devices. The automated voting system would run every time a new vote was applied to a model. It would analyze if a new voted model was acceptable for inserting into the public model. The insertion was done by finding the specific node that was being modified. Each node created within the hierarchy contains a unique identifier. When someone makes a modification to the public model they can select the entire model or a subset. Each subset is a unique node with identifier so when additions are made they can be very easily integrated back into the public model.

The major issues with the system and tools was that in-order to create a 3D building effectively there must be a substantially number of users contributing to the system. This was made apparent through attempted user studies where not enough participants were contributing and ranking different models of a building. Therefore incentives had to be developed to drive further participation and make such a tool more widespread. Unfortunately, this was out of the scope of this work.

Lastly, there was still the issue of generating a graph to use for navigating and routing. Crowdsourcing the creation of the buildings solved the issue of generating consistent, more accurate, and easy to handle 3D data. But, the graphs for navigation still had to be built by hand and done manually in a separate tool. An automated approach or technique would have made this process much easier to integrate crowd sourced models into the evacuation simulation and mobile response tools.

6.3 HURRICANE-VA

Along the same thread as evacuation planning, simulations and mobile response further effort was applied at adapting this knowledge to hurricane simulations. Ensemble visual analysis architecture with high mobility for large-scale critical infrastructure simulations [43] looked at analyzing hurricane simulations on critical infrastructure for disaster response and city planners using mobile devices. The critical infrastructure included road networks, electrical grid, communication grid, etc. The hurricane simulations were developed as an exhaustive simulation space in which as many possibilities were run to create probability of what might happen during an event. Historical hurricanes as well as simulated ones were used to collect these simulation runs. They were then combined together to create statistical measures for outages across all features.

Evaluating our system and collecting results through a traditional user study is challenging because the focus of our work is centered on both domain experts and responders. Such evaluation requires a group of responders from multiple domains working on a large collaborative disaster response exercise. Due to the limited availability of responders, I have demonstrated $HurricaneVA$ through simulated scenarios to determine our systems usability, effectiveness, and need for improvements. In this process, I conducted two evaluation sessions with three city emergency responders and four power grid managers from a large regional power company. I thus are getting feedback from both responders and domain experts.

One of our architectures advantages is the ability to compute massive amount of physically-based infrastructure simulations ahead of a hurricane season. All collaborators consider this is of great value as they can start their planning process a couple of months ahead and become fairly comfortable with the range of resiliency for their infrastructures. As summarized by one of the emergency planners, this system gives us the ability to look forward and be prepared. It could help us to find a vulnerable building and plan ahead the scenarios when a hurricane actually hits.

However, since our analysis is built upon simulations, managers from the energy company suggested further extensions to fuse heterogeneous datasets into the mobile ensemble visualization. In particular, they are interested in learning how I could effectively associate public information, such as census and demographics data.

This work was also part of additional work in understanding the impacts of natural disasters on critical infrastructure [101]. The benefit, power and ubiquity of mobile devices was made clear in the prior work on indoor mapping and mobile response. One of the major benefits of a mobile device is the fact that it can be taken anywhere. Therefore the goal of the work was to develop systems and tools to allow visual analysis of complex hurricane simulation data from a mobile device.

6.4 LIDARVA

The prior goal of Ensemble Vis was to help disaster response and city planners for hurricanes and other disaster events. Along that same thread of aiding city planners and disaster responders for these types of events was using and analyzing LIDAR data in coastal regions to assist further in finding areas of interest that might be at risk or helpful for responders [42]. Therefore, a tool was needed that was capable of doing search by example across large coastal data sets very quickly without training datasets or supervised learning. Use cases included finding large flat areas suitable for helicopter landings, buildings in at risk areas, and road networks. I tackled this problem by using image analysis. Image analysis provided the means by which to handle large amounts of data, highly parallel processing for scalability and speed, and lastly for searching. Raw LIDAR data can be very large often it is kept in different formats and coordinate spaces. This makes managing the data difficult. Converting the data to an image using lossless compression can greatly minimize the size of the data as well as retain and even add significant information. The current techniques for searching for features inside of images has been well developed. Techniques like image thresholding and blob analysis are highly effective in these areas, but the problem with thresholding is that the image must first be transformed into a state that allows effective thresholding. Even then meticulous care must be met to select values to perform thresholding.

Therefore, I developed a process to analysis the LIDAR data from a triangulated mesh and encode the analyzed information into a 2D image to use in blob extraction. I focused on using GPU computation to help solve the analysis problem and speed up computation in order to integrate this into a interactive visual analytics application. Prior work for Ensemble Visual Analysis showed that GPU processing for accelerating computation to be quite effective, but a lot of work had to be developed on the GPU computation side to make this possible. This also integrated well to using image analysis as means for using GPU computing and minimizing space and time complexity. I developed this technique into a visual analytics application that allowed users to select features and search processed LIDAR tiles very quickly finding similar features. The application further allowed users to refine their selection by narrowing the metrics they are looking for. I compare each found feature with the one that was being searched for based on size and histogram comparison. Therefore, users can find features similar in size, smaller, or larger. They can also very easily look for features that are very similar in content based on their original search.

I took some of the lessons from Ensemble Vis by further utilizing GPU computing to aid in analysis of LIDAR data, as well as the high performance rendering engine to handle rendering high resolution images and perform out of core rendering. I took the previous rendering engine from Ensemble Vis that was developed for both mobile, and laptop computers. I extended it to fully utilize desktop class GPUs and multi-threaded systems to use in a high-performance analysis application. I included features like multi-threaded off-screen rendering that would be needed for handling larger datasets like LIDAR. But most of our efforts went into developing the necessary tools for running analysis on the GPU. I developed a process for taking raw triangulated LIDAR data and converting it to a 2D image encoding with unique values to use in blob extraction. This is done by analyzing triangulated LIDAR data on the GPU for massive parallelism. Values are encoded into the RGB color spectrum for each triangle and then rendered into an image using a fixed orthographic projection.

6.4.1 CASE STUDY AND RESULTS

I ran an informal usability study evaluating the tools and systems. As part of our case study I followed previous methods described in [126]. I selected 10 participants for an informal evaluation of the tool having them complete 3 basic tasks that focus on the visual analytics portion of $LiDARVA$ The participants in our evaluation were equally balanced between male and female. Their ages ranged from college age to senior citizen age. The professional background of our participants also covered a wide range from engineering to medical professional.

I set up three tasks ordered as to how a prospective user would use the application. In the first task I focused on discovery, exploration and selection. In the second task I took the prior selection from task 1 and built upon it by asking the participant to analyze the results of their selection. This included evaluating the spatial analysis results and detailed results.

Table 4: User evaluation question result averages, ranked from 1 to 7. 1 being the worst and 7 being the best.

Question	Average
How easy was it to use navigating and zooming.	6.5
How easy was it to use the box selection tool.	6.625
How easy was it to use the lasso selection tool.	6.125
How intuitive was the scatter plot and understanding the results.	5.875
How helpful was the annotation overlay in examining results.	5.625
How helpful was the heatmap in locating results.	5.75
How easy was it to locate buildings of similar size.	6.0
How helpful is the detailed list of ranked results.	5.625
How easy was it to locate a result on the map.	6.25
How would you rate the overall ease of use of the interface.	6.0
How similar do you think the results are from a selection.	5.625
What was the most helpful function.	Heatmap

In the last task I asked participants to use the interactive tools to further drive the analysis by narrowing their original selection and analyzing the results from those interactions.

At the end of the session I followed up with participants with a questionnaire of 12 questions, 3 for each task and 3 detailing the overall usage of the tool. Each task had 3 questions that used a Likert scale to rank a particular sub-task. The final 3 questions ranked the overall learning experience and ease of use. The questionnaire also included a descriptive question asking which feature or function was most helpful to accomplish a task. The questions can be seen in table 4.

In the first task I covered navigation and zooming as well as the use of the box and lasso tool. Initially I asked participants to navigate and focus on a particular building. I located for them on the global view a particular area of interest and asked them to navigate and focus on that region. I wanted to measure how easy it was for a participant to pick up the application and navigate to an area of interest. Secondly, I had them use both the box and lasso tool to make a selection around the area of interest. These two operations are the core of initiating the analysis of our tool so being able to very quickly navigate to areas of interest and make selections is vital to the analysis. All of the participants highly ranked the ease of use for navigating and focusing on a particular region.

In the second task I had participants analyze the results of their prior selections from task 1. I first started by asking them to read from the scatter plot of results and detail for us the overall metrics of the results. Reading from the scatter plot included identifying the complete range from the smallest result returned to the largest, as well as the least similar result to the most similar. I also asked them to tell us what the mean value was for each metric. The goal of this task was to get participants to analyze the results of their selection by evaluating the results returned in the scatter plot and doing visual analysis. As part of the visual analysis I had participants use the heatmap to locate regions and then had them perform basic interaction with the annotations to find results with spatial context.

I asked participants to use the heatmap overlay to tell us where there are areas of similar features. An example of this overlay can be seen in Figure 16. I then had the participants navigate to those areas. Next, I asked participants to switch from the heatmap overlay to the annotation overlay. I then asked participants to select one of the annotations in the global view in the region they had just navigated to and examine the result.

As part of our evaluation I had participants rank the ease of use in using the scatter plot, understanding the scatter plot results, using the heatmap and interacting with the annotations. All of these subtasks were ranked highly for ease of use and understanding. The heatmap function was also listed as the most helpful function for accomplishing tasks.

In task 3 I asked participants to use their prior selection from task 1 to narrow down the search results to find buildings of similar size with the most similarity to their selection. I then asked participants to locate the top three similar features for us on the map using the detailed results table as seen in Figure 20. This final task built upon the two previous tasks by first requiring navigating and selecting a region of interest. Then, from task 2, the participant analyzed the overall results of their selection in both the scatter plot and spatially.

I started by telling participants that I wanted to find features within a specific size range with the most similarity. This required the participants to interact with the scatter plot results by filtering out all the results below and above certain values, in this case to find features of a certain size and with the most similarity.

Once an interaction occurred in the scatter plot, it is updated to the detailed results table that sorts all the results based on their similarity metric. Participants then used this table to select the top three features that were most similar. Using the detailed view they were able to then locate and point out where these features were on the LiDAR patches. Overall for all three tasks, I received strong positive rankings on the follow-up questions I asked. At the end of the questionnaire I asked three general questions about the overall system. The first question asked participants to rank the ease of use for the application. The second question asked participants how similar they thought the results were to what they had originally selected. Last, I provided an open-ended question asking participants what they thought was the most helpful feature. I received high ranks from our participants for the overall ease of use in using the interface, carrying out the tasks, and understanding the analysis results. The participants also thought they results they found were very similar to what they had originally selected.

All of the participants found the system easy to use and intuitive. They found the overall interface and interactions useful and informative. They also found the results through our searching techniques to be very similar to what they were searching for. Overall the tool was successful.

Additional work could also be done in examining better similarity metrics. I defined our similarity based on the size of the feature found to that of the original selection. This created a metric based on size. Then I used a histogram comparison between the feature that was found and the original. This created a metric based on the contents of the found feature. I used an existing histogram comparison function, but more recent comparison functions have been developed that could further improve this work. Also new metrics could be developed that provide better comparison features between selections.

6.5 STKDE

In this section, I present a GPU-based implementation of the Space-Time Kernel Density Estimation (STKDE) that provides massive speed up in analyzing spatial-temporal data. In our work I are able to achieve over 7 times the performance for the current state of the art using half the CPU resources for CPU computing, whereas on the GPU end I are able to achieve almost 100 times the performance. I have integrated this into web-based visual interactive analytics tools for analyzing spatial-temporal data. The resulting integrated visual analytics system permits new analyses of spatial-temporal data from a variety of sources. Novel, interlinked interface elements permit efficient, meaningful analyses.

6.5.1 CASE STUDY AND RESULTS

In this section, I discuss an informal usability evaluation that I ran using 10 participants evaluating the addition of our technique to a visual analytics tool. The participants in our evaluation were evenly split between male and female. Their ages ranged from college age to senior citizen age. The professional background of our participants also covered a wide range from medical to engineering. I used a Likert scale for each question, ranging from 1 (not at all) to 7 (very much so). The scores are indicated in the boxes below, where the questions in each box are on the same topic. Initially I showed the participants a multi-coordinated 2D spatial view + temporal graph. From the temporal graph, participants could select ranges in time through brushing as seen in Figure 30. Upon making a selection, a 2D kernel density estimation was generated showing the spatial hotspots in space for that selected time range.

I started with the multi-coordinated 2D spatial view and time graph as a control and a starting point before introducing the STKDE. I wanted participants to try and analyze the data to find clusters in space and time before I showed them the STKDE. In the next part of the evaluation I introduced the STKDE as an additional visualization tool alongside the 2D spatial view + temporal graph, linking it to the 2D view and the time graph. I calculated the STKDE for the entire data set. I asked participants to examine the 3D structure produced by the STKDE and to detail where clusters occurred using just the visualization of the STKDE. This was done by asking the participants to verbally state where the clusters existed geo-spatially i.e. the cluster is over Orlando, Florida. After introducing the addition of the STKDE and allowing participants to examine the structure in 3D, I asked them to interact with it further by doing temporal slicing.

All of the participants found the addition of the STKDE much more useful for finding clusters then just the 2D view as show in table 5, question 2. But, slicing was not nearly as helpful as I had anticipated and that additional slicing should be added to improve the usefulness. Therefore, after this evaluation I included spatial slicing to further improve interaction and analysis. I asked participants if they thought the calculation time would effect the overall usefulness of the STKDE. I used a comparison of 5 seconds to 5 minutes because our approach is significantly faster than prior work given the calculated speed up. Such a comparison is reflective of the time difference that one would see using both approaches. Overall they thought that the such a time difference was significant to the usefulness of the STKDE as show in table 5, question 11. Although the evaluation gave the

Table 5: User evaluation question result averages, ranked from 1 to 7. 1 being the worst and 7 being the best.

Question	Average
How intuitive was the time graph + 2D spatial view.	5.625
How easy was it to find clusters in space-time using the 2D view + time graph.	4.75
How easy is the STKDE to understand.	5.0
How easy is it to find clusters in the STKDE.	6.0
How easy was it to understand temporal slicing.	5.25
Is slicing helpful to understand certain regions in time.	5.75
How beneficial is the STKDE as an additional visualization.	6.375
Does the STKDE structure provide new information about clusters in space- time.	6.375
Was the addition of the STKDE easier then just the 2D view to find clusters.	6.25
Is the addition of the 2DView/slicing helpful to understand the STKDE.	5.375
If the STKDE took 5 minutes instead of 5 seconds to calculate would that make it less useful for these interactive tasks.	5.75

3D STKDE spatial view linked to interactive temporal slicing high marks in terms of ease of under- standing and ability to select and slice clusters of interest, it became apparent to us that further interface elements would be needed for detailed understanding of the space-time behavior. Therefore, I added further interaction to the 3D view building on the responses I got from the evaluation.

6.5.2 USE CASES

I also evaluated three use-case studies I performed using our approach. In the first use-case, I demonstrate the capability of our approach to the Dengue fever dataset. In the second case, I apply our approach to analyzing hurricane simulation data. In the third use-case, I examine data provided by a local company with whom I collaborated.

DENGUE FEVER: As part of our results I compared our algorithm to that of prior work on Dengue fever outbreaks in Cali, Columbia [60]. The dengue fever dataset is important because it is based on real world cases of dengue fever. Analyzing disease spread is very much spatiotemporal in nature. The data itself occurred during the first six months of the year. It can be difficult to understand how disease cases migrate to different areas, but also how they evolve over time. Using the STKDE reveals this behavior efficiently and effectively. In Figure 41 I show four different perspectives of the resulting structures.

The structures show very clearly how certain areas were affected over time. The STKDE produced six unique structures. The majority of the clusters begin closer to the start of the year and then gradually decline towards the end. There are two larger elongated, spatially localized structures side by side that begin very early on and continue for a significant amount of time. These indicated locations where the disease persisted. These clusters also occurred in highly populated areas. Interestingly, there are several smaller structures that begin at different times and then very quickly stop. These indicate areas where the disease spread to. They also show a pattern of how the disease moves north through the city. Our interactive interface permitted study of the relationship of all these patterns plus detailed examination.

HURRICANE DIANA: In prior work [79], I analyzed Hurricane simulation data for critical infrastructure in the state of North Carolina. This simulation data contained the effects and outages caused by hurricanes across multiple simulation runs in order to build an exhaustive simulation space. The interconnected infrastructures include multiple systems like electrical systems, road networks, transportation, water and sewage networks. These simulations also include cascading events. For example a hospital might lose electrical power due to the outage of a substation, but would continue to function because of electrical generators. At some point in the simulation those generators would fail causing the hospital to become disabled. It can be hard to visualize these types of cascading events so that one can discover and understand their relationships in space-time. Our simulation data is very much spatiotemporal for these reasons. Therefore, including the space-time kernel density estimation is an important addition for understanding these types of events. In prior visualization work [43], I focused on a mobile application for first responders and planners to use in the field. The tool used complex filtering on different categories of critical infrastructure, hurricane paths, and temporal filtering. I then used these filters to generate 2D spatial hotspots in real-time. For these cases pre-computing space-time behavior was near impossible because the combinations of filters was exponentially large. Our STKDE is a great addition because it is fast enough for dynamic filtering thus making possible detailed analysis of cascading events and what causes them. Using the STKDE revealed some important structures that were previously hard to see.

For example, Hurricane Diana hit the coast of North Carolina, moved back out into the Atlantic Ocean, and then circled again inland. The resulting space-time structures clearly reveal the relationships and differences between the first and second hits over space-time as shown in Figure 42. I see the overlapping effects of the two hurricane passes, where they start in time, and how they evolve. I see that the areas of infrastructure breakdown follow the hurricane path to some extent but then spread on their own. It would be difficult to study this behavior in detail without the STKDE.

DEVICE REPORTING DATA: I collaborated with a local company to acquire some of their data to evaluate our approach. The company I worked with has numerous interactive devices all across the United States. These devices range from interactive kiosks to point of sale systems in retail locations. Hundreds to thousands of users interact with and use these devices daily generating large amounts of reporting data. I acquired a small subset of that data consisting of approximately 1,000,000 records. The records covered a time period of one month. I met with the Chief Technology Officer (CTO), President of Technology, Director of Operations, and President of Sales and Marketing, using our techniques and tool to evaluate the data they gave us and elicit their feedback and analysis. Initially I showed them a 2D view + temporal graph as I did in our user evaluation. I did this initially because I wanted them to think about the results they saw just using a 2D view. This way they would develop their own ideas about where hotspots occurred in space and time. I examined several of their retail locations across the East Coast using the 2D view trying to make sense of where hotspots occur in space-time.

In the next step I included the addition of the STKDE and it immediately brought to light where hotspots occurred. I followed the same procedure and analyzed the same groups of retail locations as I did using just the 2D view. I found several interesting examples using the STKDE that would have been extremely difficult to detect without. There were several different retail locations that were examined and they all followed a cyclical pattern of peak activity on Saturdays. Some locations the activity had abnormal and interesting behavior. Locations around Virginia only peaked once. In North Carolina locations peaked then gradually declined and then stopped. Activity picked up the following weekend. However, some locations in Florida experienced very sporadic activity. All this could be visualized directly and immediately through the STKDE leading to areas for further analysis, as show in Figure 43. This can be done by setting the spatial and temporal slicing planes in the overview to see relationships among features or zooming in or particular features and then setting slicing planes. I asked the executives for a concise summation of what they thought after seeing the analysis with and without the STKDE. These were the comments they gave us.

- The STKDE adds value by adding another dimension, which provides more information.
- The human has to remember the details of the 2D and time graph. Whereas the STKDE reveals everything.
- After viewing both approaches, it is meaningless to not use the STKDE.

The data they provided to us was multivariate and contained additional information with respect to their devices, the nature of the report, software versions, and application types. They wanted to be able to run more complex filters and sub-filtering for these attributes in order to analyze using the STKDE. This highlights the major problem I worked to solve. Since the analysis is user driven, the STKDE has to be done in real-time. There are too many possible combinations to do exhaustive pre-computing for their data and their data is constantly growing. Therefore, our technique is crucial in that it solves this problem and allows for real-time analytics.

6.6 STKDE FOR TABLET DEVICES

In this section, I present a GPU-based implementation of the Space-Time Kernel Density Estimation (STKDE) that provides massive speed up in analyzing spatial-temporal data. In our work I are able to achieve over 7 times the performance for the current state of the art using half the CPU resources for CPU computing, whereas on the GPU end I are able to achieve almost 100 times the performance. I have integrated this into web-based visual interactive analytics tools for analyzing spatial-temporal data. The resulting integrated visual analytics system permits new analyses of spatial-temporal data from a variety of sources. Novel, interlinked interface elements permit efficient, meaningful analyses.

To evaluate our visual analytics system, I conducted a formal user study. Our IRB approval number was 16-0941. The study is a between group study with the goal of evaluating an expert group with background in the STKDE, but minimal experience in visual analytics on tablet devices. The purpose of evaluating this group is to see if our tablet based interface is easy and effective to use for someone knowledgeable with the STKDE, but unfamiliar with visual analytics on smaller devices with touch input.

I also evaluated a second group who had a broader range of knowledge of visual analytics, interaction and tablet based interfaces, but was very unfamiliar with the STKDE. The goal of evaluating this group was to see if incorporating the STKDE was an effective tool for someone unfamiliar with the technique, but very capable in using visualizations and tablets.

The expert group familiar with the STKDE was from Geographic Information Sciences, and the novice group was from Computer Science Visualization. I refer to the group from GIS as experts because they have extensive experience in analyzing, interacting with and understanding spatio-temporal data using the STKDE technique. They also have prior experience in industry standard tools used for analyzing spatio-temporal data, but lack any experience outside of those tools. Secondly, experts are extremely difficult to organize and evaluate in larger numbers, so it is important to identify a group like this with sufficient expertise that is also available. A main focus was to do a comparative analysis between the experts and a group of non-experts from CS who are technically savvy. The latter are often employed for user studies and evaluations because of their availability, but their performance and take-aways may be different than those from experts, and these differ- ences may be important in complex analysis applications. The tools capabilities were designed with both groups in mind. The visualization of and interaction with the STKDE were developed after consultation with a Geography professor (as was the application) who has worked extensively with non-interactive STKDE analyses. The coupling of interactions with visualization was designed based on previous studies with the second group. Ultimately, one would hope to come up with a tool that serves both groups well.

6.6.1 CASE STUDY AND RESULTS

I recruited 12 participants from GIS and CS (4 are master students, 7 are Ph.D. and 1 is a post-doc). The participants were between the ages of 21 and 54. Of these, 5 were from GIS and 7 from CS. All participants had some familiarity with multi-touch and tablet devices. All successfully finished the study within 30 minutes. The participants were provided approximately 5 minutes of training. Since our tool has no direct comparison to existing tools, I asked participants to detail what tools they currently use for analyzing spatio-temporal data and the most important feature they use from those tools. In addition, I asked all of our participants to detail their experience with multi-touch mobile devices.

The GIS experts were very familiar with tools like ArcGIS and Voxler. They cared most about visualizing spatio-temporal data through a space-time cube, volumetric rendering, and kernel density estimations. In contrast, the CS participants had little to no knowledge of tools like ArcGIS or Voxler, but were aware of visualizing spatio-temporal data using toolkits like D3. I asked all participants to perform several tasks and then followed up with a questionnaire. Prior to each task, I provided a short period of training (approximately 2 minutes) to demonstrate how to use specific features with the interface. Then I asked participants to perform the following tasks:

T1 : Draw a hypothetical hurricane path and then analyze the temporal and spatial trends in the 2D view.

T2 : Navigate the 3D view, using volumetric rendering visualization and interact with the STKDE through direct selection.

T3 : Perform temporal and spatial slicing on the STKDE and analyze the results produced by each method.

At the end of the evaluation, I asked some open-ended questions to all the participants

- How does this tool compare to any other tools that you have used?
- Could you see other applications of this tool for mobile devices?

- What improvements would you like to see added to this tool?

GIS: Overall the GIS participants were extremely impressed with the tool. When I asked them how the tool compares to any other tool they have used, the first thing they noted was the smaller screen size and lack of mouse interaction. They were accustomed to larger screens and mouse-based input, but once they learned the interaction on the tablet they expressed that it was surprisingly easy to use. They also said that even though they do not use tablet-based applications they thought it was extremely useful for disaster prep and management. They also said that our tool was more intuitive than Voxler. Here are some direct quotes:

- "Smaller screen size, no mouse. I'm use to a bigger screen and mouse-interaction. However, interaction on the tablet is surprisingly easy once you know the gestures."
- "It is more intuitive than Voxler."
- "I do not use tablet-based applications but this is extremely useful for disaster prep and management."

The GIS group has extensive experience in using the STKDE, but majority of their feedback related to the differences between using a desktop application versus a tablet device due to the unfamiliarity with visual analytics on a tablet. But, once they learned the interaction through minimal training they were pleasantly surprised how easy it was to use. This can be seen in their task-based results in Figure 45.

Lastly, I asked them what improvements they would like to see. Overall they wanted to see more interaction specific to the STKDE by being able to adjust isovalues for the isovolumes and thresholds for the volumetric rendering. They also noted that they felt using touch was in-precise in some cases and improving that would be helpful.

CS: The CS group had a much broader range of knowledge in visual analytics as well as in using visualizations on tablets. But, they lacked knowledge of the STKDE technique. Their responses differ significantly from the GIS group who reflected almost entirely on the tablet view size and the touch input.

The CS group responded more so to the interaction and their ability to understand geo-spatial events. They expressed a familiarity with 3D navigation because of its similarity to Google earth and Google maps. They thought temporal slicing on the STKDE was very effective. Overall they thought using the STKDE made it easier to understand geo-spatial events, while also allowing the user to dig deeper into the data. Following are some quotes:

- ”Very useful in its ability to estimate outages geo-spatially and temporally”.
- ”Temporal slicing is a great interaction technique.”
- ”Easier to understand information being visualized. Also gives more types of visualizations the more information you get. (other ways to see more info). Doesn’t throw numbers at you until you get deeper into the data. Not too much information all at once.”
- ”Combines 3D and 2D smoothly, specific to a problem. It’s on a tablet”.
- ”The views are very well-tuned to the particular analysis task, so the application is superior. It seems to be implemented optimally and performance is competitive. The insight is easily obtained.”

Lastly, I asked them what improvements they would like to see. They wanted to see the inclusion of additional data in order to view the cost in both human life and monetary value due to outages of critical infrastructure.

Discussion

The participants from each group were asked to answer the usability questions in Figure 44 on a seven point Likert scale where 1 was very negative and 7 was very positive. The results are compiled in Figure 45. It can be seen that the GIS and CS groups grade usability mostly positive across questions though the GIS group tends to grade somewhat lower than the CS group. However, none of the differences between groups in any of the questions is statistically significant according to p values from Friedman's test (with $\alpha = .05$ level of significance). The two groups also performed similarly in that they could do all the required tasks, completing them all within approximately 30 minutes.

The conclusion from all these evaluations is that both groups performed about the same. This is in itself a significant finding in that it shows that the STKDE and interface elements should be widely effective for this kind of application.

However, the trend shown for the usability questions would be worth studying with additional experiments. In addition to the unfamiliarity with tablet interactions, the expert GIS group might tend to grade lower because they expect more quantitative and qualitative details out of the STKDE and spatial-temporal hotspot renderings, based on their knowledge of how these results are used.

Although both groups performed similarly, their answers to the open-ended questions showed differences related to their respective backgrounds. They mentioned different new applications, of course. But in addition the expert GIS group was especially impressed with the functionality and ease of use. They noted that even with minimal tablet experience once you learn the gestures the tool was remarkably easy to use. It was even noted as being more intuitive than Voxler.

Overall our user study proves two points. First, in giving our interface to a group inexperienced with visual analytics and interaction on tablets, but very knowledgeable about the STKDE they were still able to perform well. This shows that the interface was designed effectively. Second, by giving our tool to a group with broader experience in visual analytics and interaction, but very unfamiliar with the STKDE, I showed they were still able to perform well. This demonstrates that the STKDE is effective on a tablet-based device.

6.7 HURRICANE SIMULATION EVALUATION

One of the critical missing pieces of our hurricane simulation work is evaluating the results and expanding the model to incorporate addition parameters. We need to be confident in our simulation analysis based on ground truth results from historical hurricanes. This is necessary in order to disseminate our work to city planners and first responders. But, evaluating this work is a massive undertaking and would require extensive data collection and analysis. In this section I focus on initial steps in evaluating one historic hurricane path Hurricane Matthew.

Hurricanes are extremely complex weather phenomenon that incorporate many factors. In order to evaluate our simulations I tackle each piece individually. Our simulation model currently incorporates wind speed, but lacks any parameters that take into account flooding and storm surge.

6.7.1 STORM SURGE AND FLOODING

Storm surge is an important aspect especially for coastal areas because it can cause significant and even deadly flooding. But, flooding can still occur further inland due to excessive rain brough by a Hurricane.

Therefore, these are important parameters to incorporate into our model. But generating this data is non-trivial, therefore I developed a streamlined process whereby I generated elevation data and coastal distance data for each feature. Elevation data is important because features that are at lower elevation closer to sea-level can be more prone to flooding especially in coastal regions. Therefore, it is important to take this into account.

In order to generate elevation data I used U.S Geological Survey (USGS) to do a longitude, latitude lookup of each feature in our infrastructure database. USGS provides a web-service whereby they return the elevation in feet. This data is then included for each feature in our database.

Elevation data is not enough to determine storm surge. The second piece of information is the straight line distance to the coast, but computing this is also non-trivial. To approximate stormsurge I use the straight line distance to nearest water source connected to the ocean. This is detailed as an important metric for estimating storm surge along the Florida coast line [102]. Although there are a significant number of factors that play a role in storm surge we are developing a simplistic approach first to estimate storm surge damage.

The first step of computing this data was collecting a detailed shapefile of the coast line for the USA. The shapefile contains very detailed rivers and estuaries. This is important because storm surge could flood those bodies of water as they are connected to the ocean. The distance is then computed in miles and the spatial location of the straight line is also saved for each feature in our infrastructure database.

6.7.2 QUALITATIVE EVALUATION

Evaluating our hurricane simulation effort is a daunting task. This initial effort attempts to do some qualitative analysis but more data needs to be collected to effectively analyze our work. In the first subsection I look at electrical outages produced by Hurricane Matthew and compare the overall outages to what was produced by our simulation.

6.7.2.1 OUTAGES

Power outages was the first result I examined. I was able to collect high level outage information from Florida during Hurricane Matthew. At the height of Hurricane Matthew's impact, more than 1.1 million people were without power across the east coast. The outage data was per county and reported the total number of customers without power. Using this metric it is hard to compare directly to our simulation results. Since our simulation only takes into account critical infrastructure like hospitals, substations, etc. it is very hard to do a direct comparison. But, we can qualitative evaluate the outages in each county to examine if there are unsually inconsistencies between the number of customers affected and the critical infrastructure disabled.

To compute the outages produced by our simulation per county I first collected shapefiles for each county in Florida. The outages from our simulation refer to critical infrastructure that was disabled from our ensemble simulations. For example if a substation was disabled that counts as 1 outage. I then aggregate these per county. I used a spatial database to compute if a specific feature was contained within a county boundary.

More than half of the outages are in Jacksonville - Duval County. The second largest outage was in Volusia county with almost 47000 reported outages. This can be seen in the table below. While directly comparing these outages is not feasible some things do come to light. The first thing to note about this data is that the population of each county is heavily imbalanced. Palm Beach, Orange, Dubal , Brevard, Volusia are some of the most populated counties in the state. Therfore the ratio of reported customer outages to criticial infrastructure outages may be heavily imbalanced.

One area that we noticed significant differences is in Brevard and Orange County. These counties cover the Orlando area and are heavily populated, but the reported outages is

Table 6: The number of reported people without power by county for Hurricane Matthew.

County	Reported	Simulation
Alachua	22	9
Baker	20	0
Bradford	50	69
Brevard	5770	792
Clay	1539	62
Duval	91980	228
Flagler	17526	146
Glades	0	13
Hendry	0	32
Highlands	0	68
Indian River	1870	242
Lake	82	74
Marion	4	1
Martin	150	70
Nassau	608	16
Okeechobee	0	87
Orange	70	617
Osceola	0	133
Palm Beach	0	18
Polk	0	12
Putnam	2415	29
Seminole	2130	218
St. Johns	3390	237
St. Lucie	1220	229
Union	0	24
Volusia	46937	578
Total	175933	4074

significantly different. Therefore, there is some additional improvement that could be made to the model. One explanation for this difference could be that the infrastructure in the Orlando area has already been reinforced since this area is heavily populated and is in a prime hurricane location.

Lastly, while this is an initial evaluation additional data is necessary to fully quantify the differences

6.7.2.2 AERIAL IMAGERY

In this last section we look to aerial imagery and LiDAR data for a more quantitative analysis. Acquiring this data is difficult because of the temporal component of when it needs to be collected. Some data has been collected by NASA, but it is limited in scope. The work I've done on LiDAR analysis is a great application for analyzing the differences before and after a hurricane. Unfortunately, I was unable to acquire LiDAR data from hurricane Matthew, but I was able to collect some initial aerial footage.

In this initial work I was able to apply the semantic segmentation model to the aerial images. The original model was trained on triangulated LiDAR data that was then rendered to an image. This model worked well for LiDAR data, but it's accuracy is not as high for aerial images. To use aerial images they are first converted to gray scale. An example of the before segmentation can be seen in Figure 46. This example shows an airport in Greenville, NC that has been flooded by Hurricane Matthew. The areas segmented in pink are roads, the areas segmented in yellow are high vegetation like trees, and the areas in blue are buildings. The after picture and analysis can be seen in Figure 47. Clearly, there are areas of the runway that have been flooded causing the airport to be disabled.

From our simulation results we were able to located this particular airport and it was not disabled in our simulation. Since our simulation does not take into account flooding we would not be able to reproduce this result. Therefore, we can examine results like this to include better flooding approximations given the previous elevation and costal distance data we've collected.

There are several improvements that can be made to this analysis. Firstly, the model itself can be improved. It was originally trained on LiDAR data but could be modified to operate on aerial imagery. Unfortunately it is unlikely that LiDAR data was collected for Hurricane Matthew as any damage has long past so we must look to aerial imagery for now. Secondly, this analysis step could be automated whereby the segmentation pieces from the before and after images could be compared and then the results evaluated from the simulation.

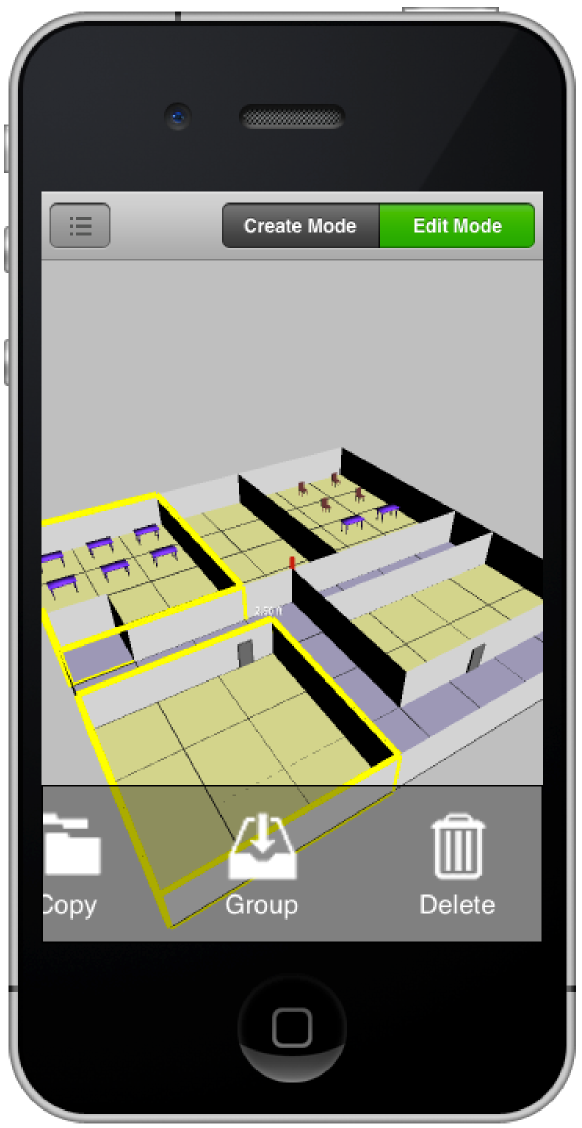


Figure 40: Example interface for 3D modeling By the Masses iPhone application. It features a simple interface for editing 3D buildings. It includes all the basic features of a 3D editor to copy objects and group them into complex structures.

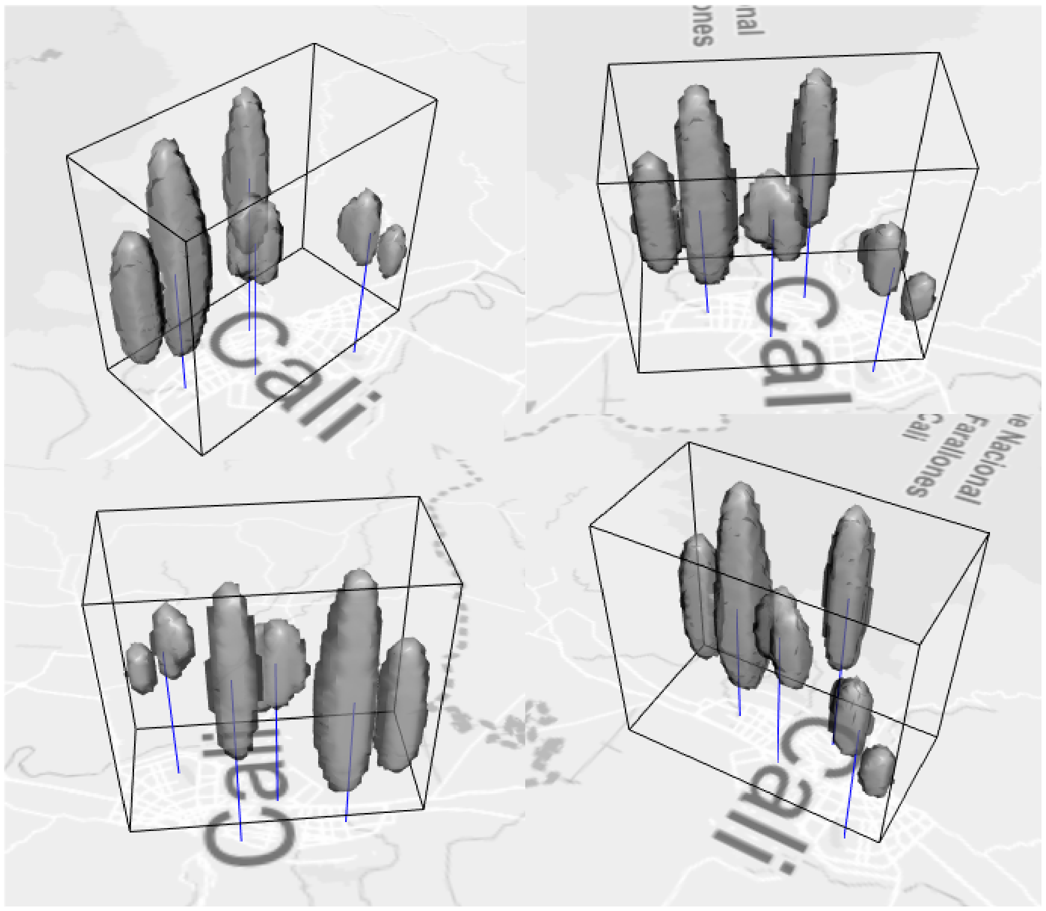


Figure 41: Space-Time Kernel Density Estimation Dengue Fever Use Case.

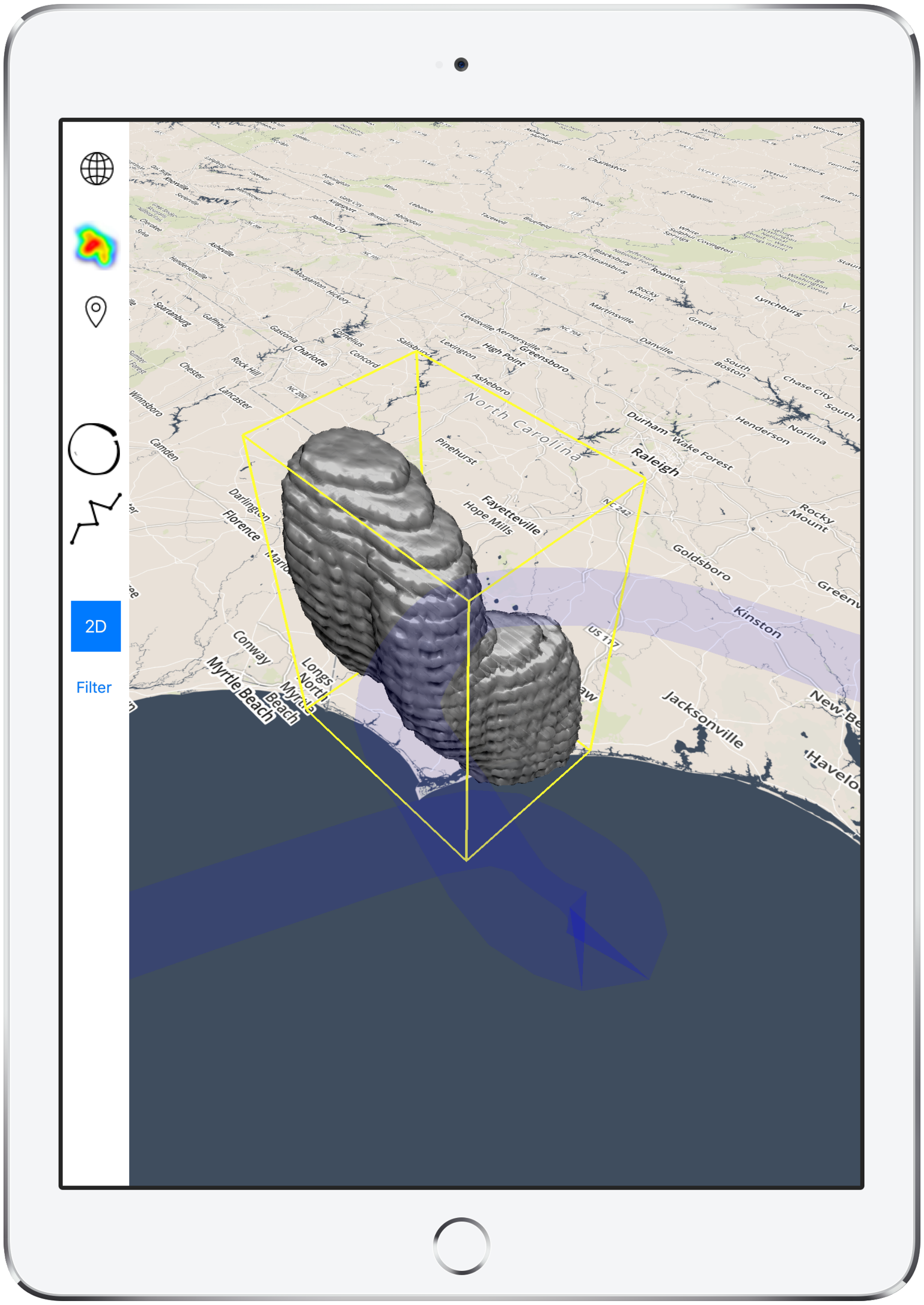


Figure 42: Space-Time Kernel Density Estimation Hurricane Diana Use Case.

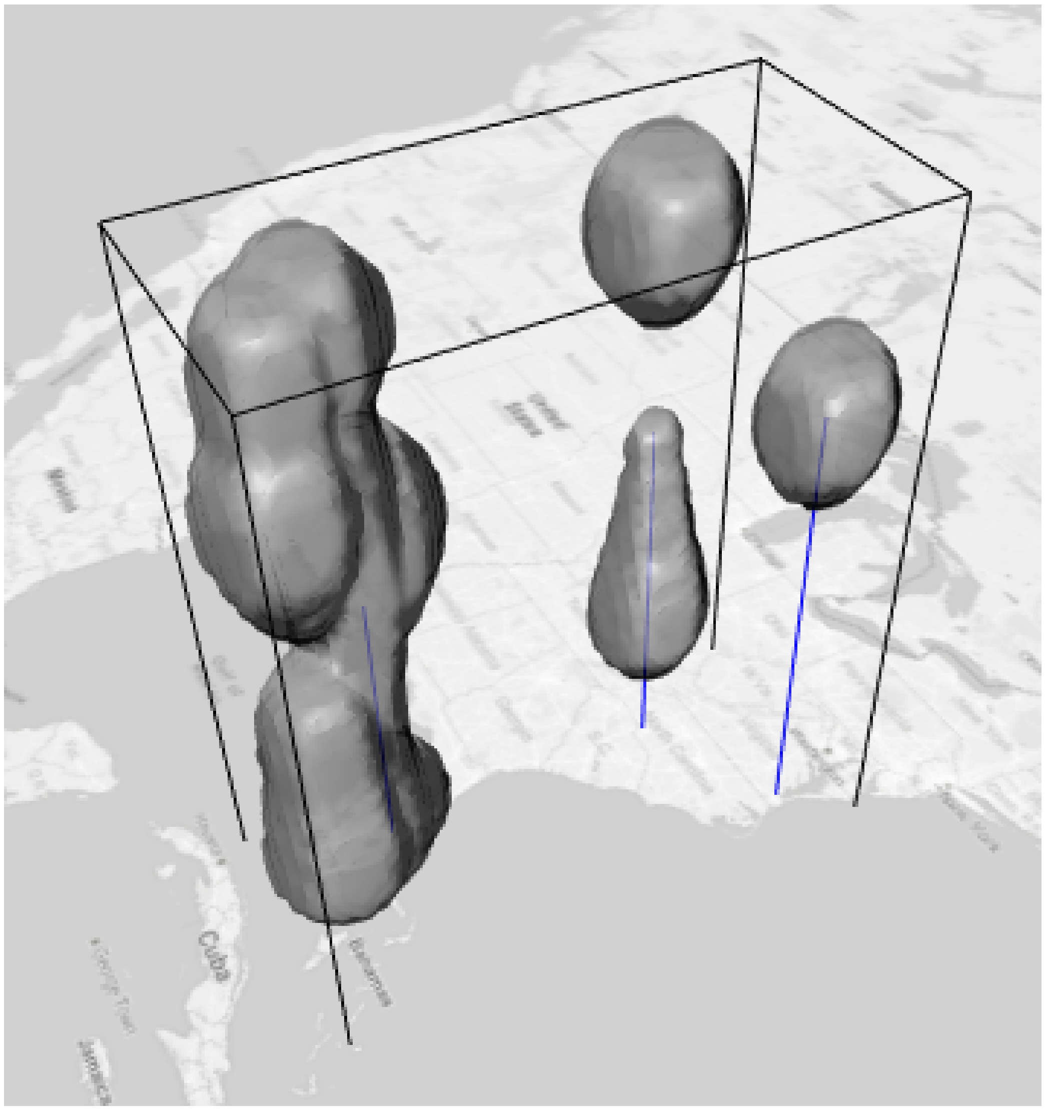
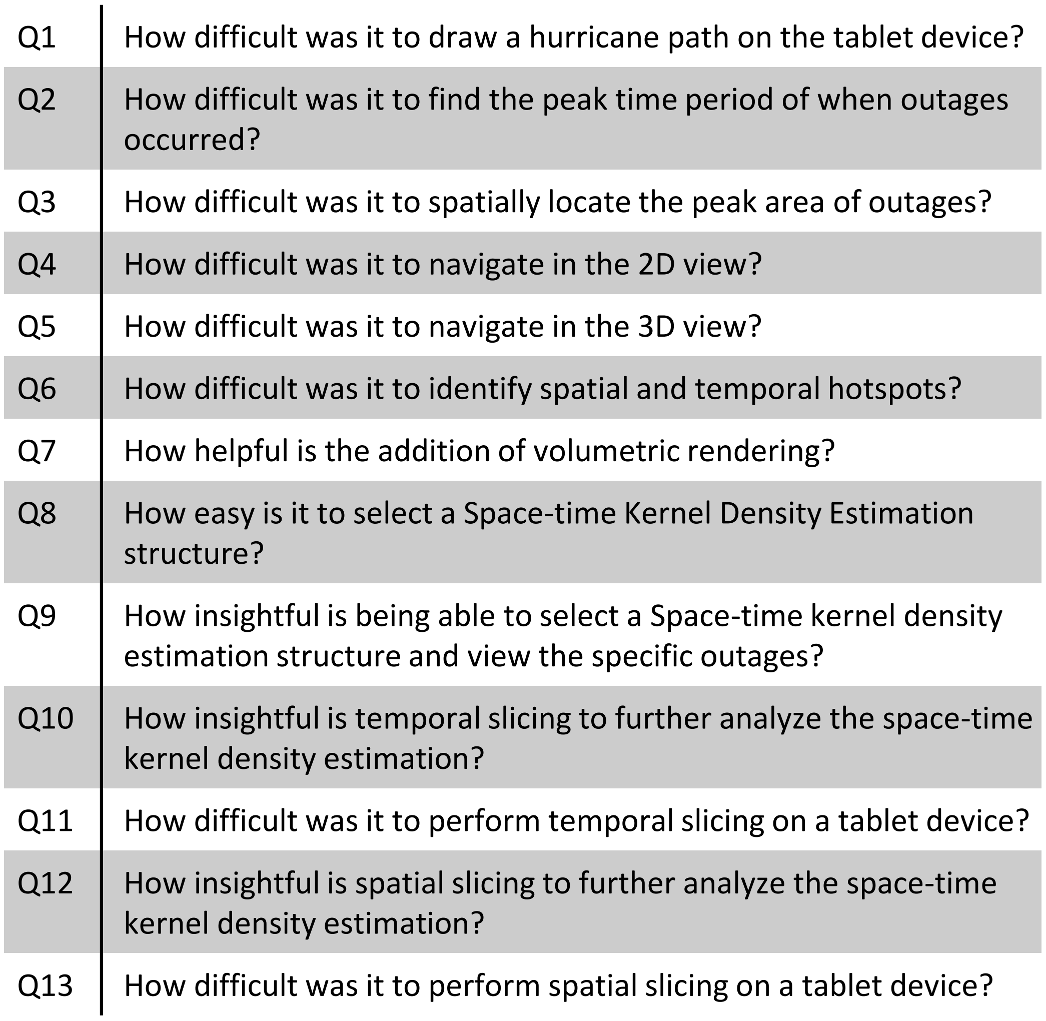


Figure 43: Space-Time Kernel Density Estimation Device Reporting Data Use Case.



Q1	How difficult was it to draw a hurricane path on the tablet device?
Q2	How difficult was it to find the peak time period of when outages occurred?
Q3	How difficult was it to spatially locate the peak area of outages?
Q4	How difficult was it to navigate in the 2D view?
Q5	How difficult was it to navigate in the 3D view?
Q6	How difficult was it to identify spatial and temporal hotspots?
Q7	How helpful is the addition of volumetric rendering?
Q8	How easy is it to select a Space-time Kernel Density Estimation structure?
Q9	How insightful is being able to select a Space-time kernel density estimation structure and view the specific outages?
Q10	How insightful is temporal slicing to further analyze the space-time kernel density estimation?
Q11	How difficult was it to perform temporal slicing on a tablet device?
Q12	How insightful is spatial slicing to further analyze the space-time kernel density estimation?
Q13	How difficult was it to perform spatial slicing on a tablet device?

Figure 44: Usability questions.

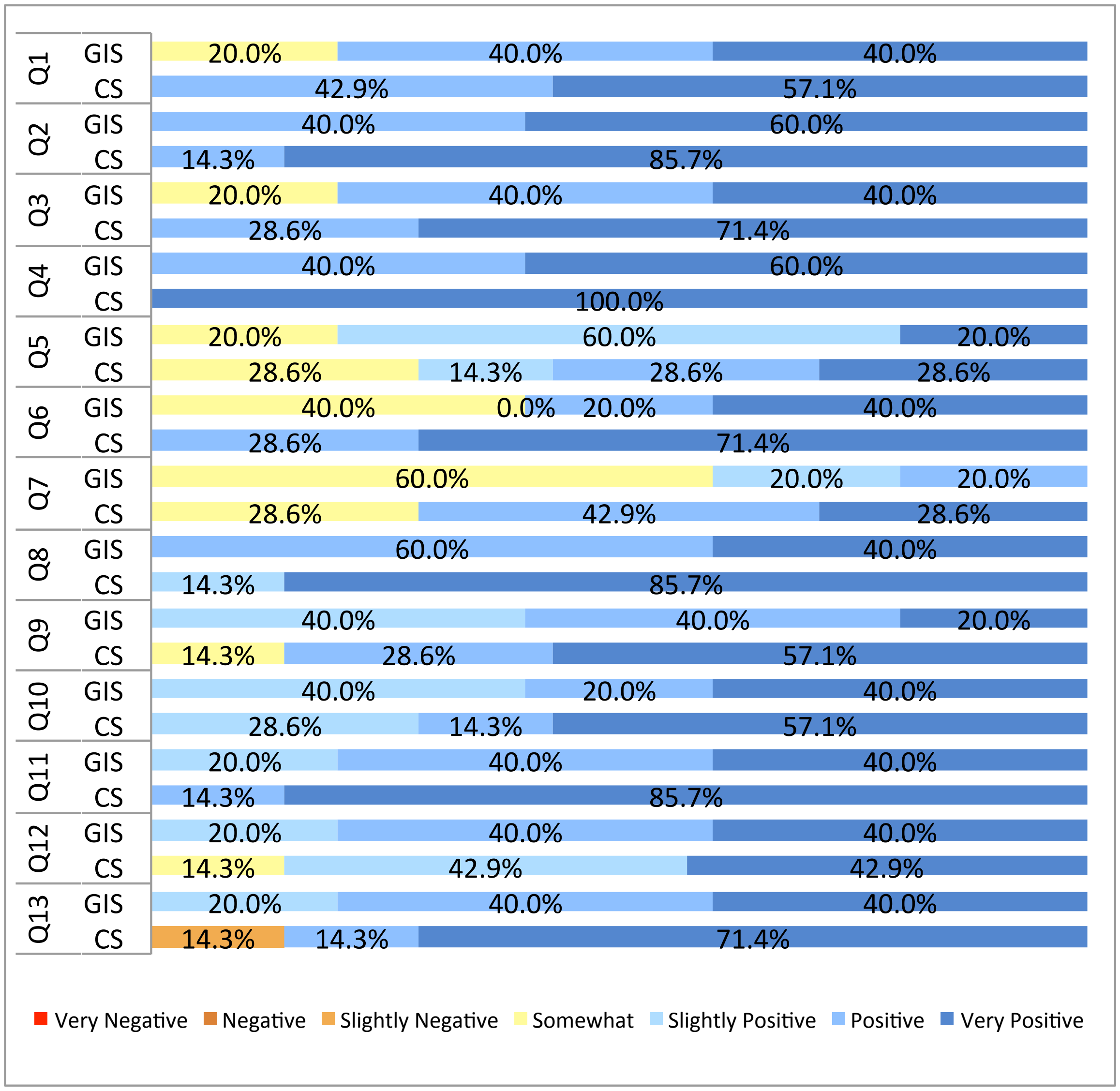
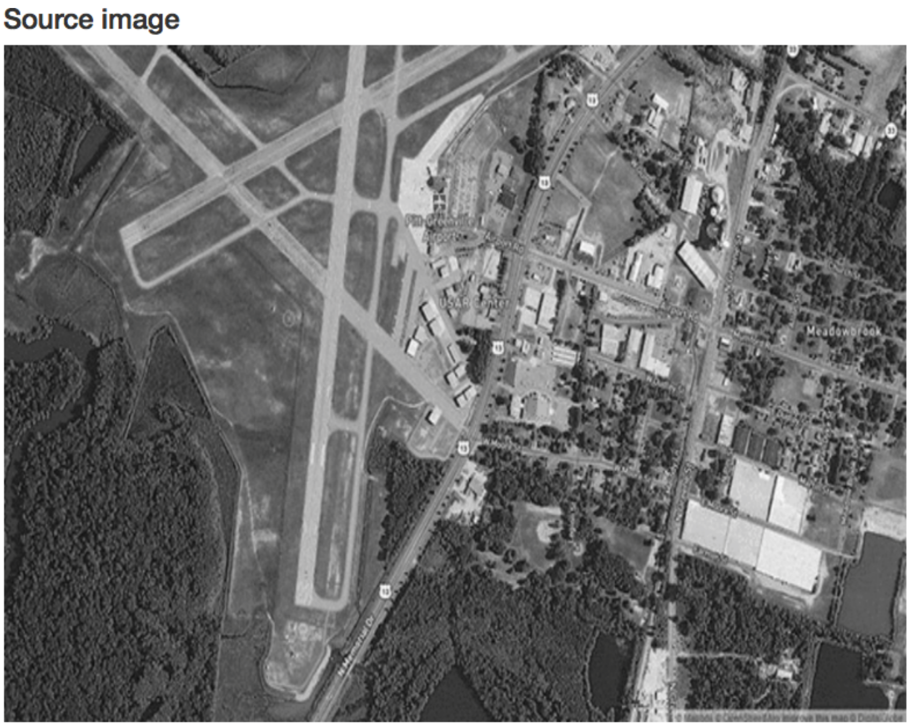


Figure 45: The combined results of the responses from each participant group. Each question is broken down as a total percentage of how many participants in that specific group answered very positive, very negative or in between.



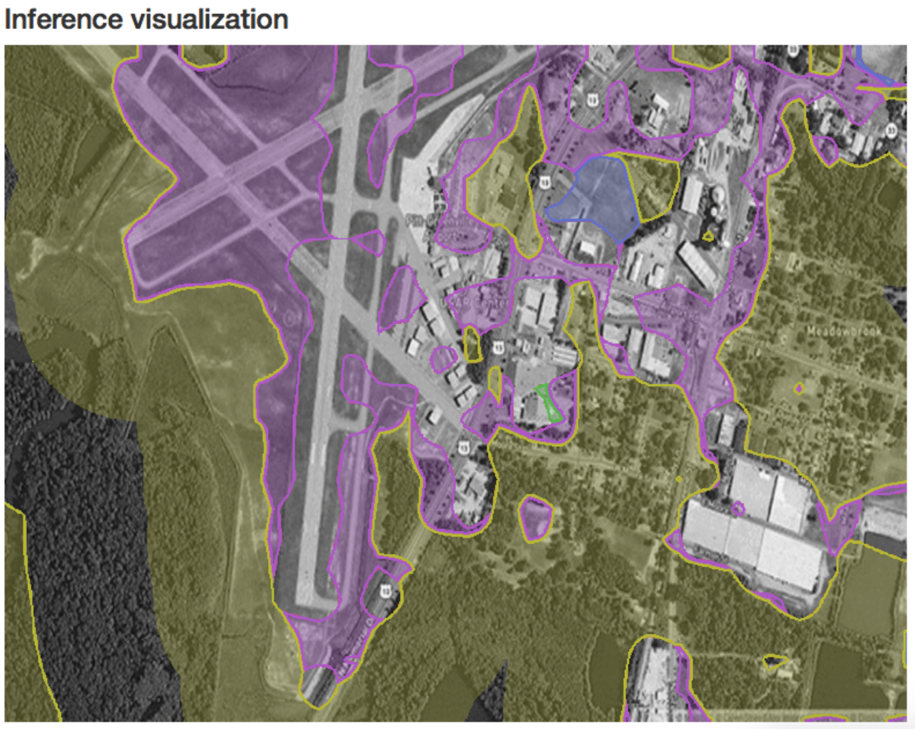


Figure 46: The before image and segmentation results of Pitts-Greenville airport in North Carolina.

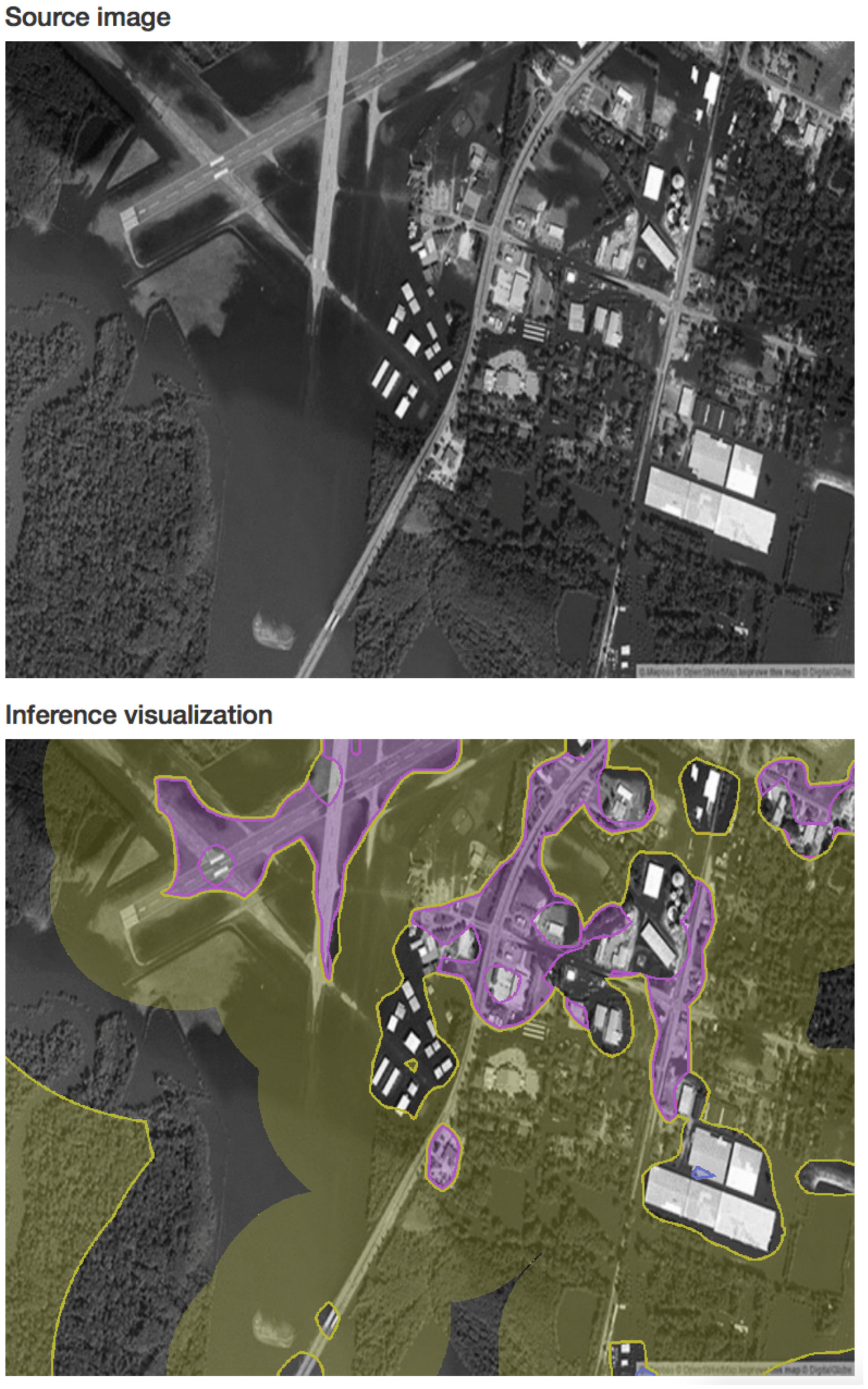


Figure 47: The after image and segmentation results of Pitts-Greenville airport in North Carolina, which shows substantially flooding of the runway.

CHAPTER 7: CONCLUSION

In this chapter I recap and summarize all the work completed for this dissertation. The challenge of Big Data is multifaceted with the end goal of extracting actionable knowledge. Big data is a powerful tool in today's world, because decision makers can use it to garner knowledge, make critical decisions and optimize existing processes. But unlocking the potential of big data first requires us to solve a number of challenges. It requires the integration of techniques, technologies, and visualizations to uncover hidden meaning from datasets that are diverse, complex, and of a massive scale. Big data is beyond the ability of commodity software tools to manage, process, and visualize in a reasonable amount of time. Therefore, new techniques, technologies, and applications must be developed in order to provide scalable, situational awareness for decision makers [18, 66].

Data and Analytics: In the first chapter I discussed data and analytics. The challenge of big data is a difficult problem. There exists lots of different types of data that can be unstructured, heterogenous, and multi-variate in nature [135]. This poses a significant challenge because there are so many different types and each type might entail a unique problem that needs to be solved. In order to solve these problems patterns and meaning have to be extracted before actionable knowledge can be made. Therefore, different strategies have to be developed as the two are very deeply inter-linked. These areas include spatio-temporal, graphs and networks, remote sensing, crowd source, social media, and simulations.

For my dissertation I focused on data with respect to several domains:

- Emergency Response and Evacuation Simulations.

- Hurricane Simulations on Critical Infrastructure.
- Remote Sensing.
- Social Media.

In our work on emergency response and evacuation simulations we developed analytical techniques for handling indoor routing and evacuation simulations. In my work on Hurricane simulations on critical infrastructure I developed methods for handling spatio-temporal data, specifically using the STKDE.

I also developed a novel analytical process for analyzing LiDAR data from remote sensing. We also developed new methods for analyzing hetergenous data types from social media to identify health related problems in college students.

Visualization: Visualization is an important aspect in understanding the value of big data. Without visualization it is impossible to discern meaning in order to make well informed decisions. The field of visualization is extensive and covers many disciplines. In this dissertation I detail several tools developed and the visualization aspects within those tools. These tools included evacuation simulations, emergency response, remote sensing, spatio-temporal using the STKDE, and hurricane simulations.

An important part of developing visualization tools is evaluating them through use-cases or usability testing. Each of these tools were then evaluated in some way to show the effectiveness of the visualization.

Scale: One of the significant challenges of big data that I tackled in this dissertation is the sheer volume of data. I focused on the use of cloud computing to serve numerous types of clients, systems and platforms. The second major focus is the utilization of parallel computing. Since majority of the data and analytical computation requires siginifcant hardware resources it is not possible to perform these types of analysis at scale on mobile and commodity devices. Therefore, specialized algorithms have to be developed.

As my major contribution to scale in this dissertation I developed two novel processes for utilizing GPU computing to analyze and handle data. The first focused on remote sensing, specifically LiDAR data, whereby I utilized GPU computing to analyze LiDAR at massive scale. The second major contribution was the acceleration of the STKDE to utilize GPU computing. In that work I developed a novel GPU based algorithm that provided massive speed up and sub-second computation time for datasets transmittable over the internet.

Platform and Interaction: Platform is an important piece to situational awareness and visual analytics because it enables users to do significantly different things with knowledge acquired through visual analytics. Once I have developed strategies for analyzing and visualizing data at scale I then have to choose the best means for presenting and allowing users to extract meaning and usefulness through a platform and tailored interaction. Interaction can be unique to the individual platform because mobile devices rely on touch based gestures. So complex multi-coordinate views that require a mouse must be redesigned and developed for use on a mobile device.

I detailed several tools that were developed as part of this dissertation that utilized web-based and mobile platforms. I discussed design elements specific to each platform and how visualization and interaction must be thoughtfully adapted in order to provide effectiveness and ultimately insight into data.

One specific type of interaction that I also focus on as our contribution is search by example. Search-by-example often involves some type of data analysis. While search by example for image based data is a well researched field I focus on the novelty of applying it specifically to LiDAR data. As well as evaluating a hypothetical hurricane path in real-time, in terms of relevant results from our ensemble simulation collection. This provides immediate, actionable results to emergency planners and first responders.

Evaluation: Evaluating our work is the most important aspect of understanding the effectiveness of our tools and processes. For each of the tools that were developed they were also evaluated through use-cases and informal and formal user evaluations.

We evaluated our emergency response and evacuation simulation tools directly with UNC Charlotte campus police. By evaluating these tools directly with campus police and solicting feedback from officers and the police chief, we were able to fully understand their needs and how our tools can better fit into their protocol. One important need stressed by the police chief was the necessity for mobility. Being able to take tools with them into the field was absolutely crucial. Therefore, this drove our future work in a mobile direction focusing on flexible and versatile platforms.

My work on remote sensing and the STKDE was informally evaluated through usability testing and use-cases. Evaluating the usability was important in that it demonstrated the ease of use and overall usefulness of each tool. It was also crucial for understanding the effectiveness of the STKDE as a visual analytics tool.

Lastly, I put together a formal evaluation of the STKDE for tablet devices, which was a culmination of the work I had done as it encompassed every aspect of my dissertation. As part of the evaluation I organized two different groups with significantly different backgrounds and expertise. Overall the evaluation showed the ease of use and effectiveness of the STKDE on a tablet device. Ultimately the use-cases and evaluations performed showed the effectiveness, insight, and powerful capability of our tools. They demonstrated that we were able to solve the problems orginally outlined in this dissertation.

REFERENCES

[1] D. Agrawal, S. Das, and A. El Abbadi. Big data and cloud computing: Current state and future opportunities. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 530–533, New York, NY, USA, 2011. ACM.

[2] A. Akaydın and U. Güdükbay. Adaptive Grids: An Image-based Approach to Generate Navigation Meshes. *Optical Engineering*, 52(2):Article No. 027002, 12 pages, February 2013.

[3] N. Andrienko and G. Andrienko. *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[4] O. K.-C. Au, C.-L. Tai, and H. Fu. Multitouch gestures for constrained transformation of 3d objects. *Computer Graphics Forum*, 31(2pt3):651–660, 2012.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[6] T. Blaschke. Object based image analysis for remote sensing. {*ISPRS*} *Journal of Photogrammetry and Remote Sensing*, 65(1):2 – 16, 2010.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[8] K. Blumenstein, C. Niederer, M. Wagner, G. Schmiedl, A. Rind, and W. Aigner. Evaluating information visualization on mobile devices: Gaps and challenges in the empirical evaluation design space. In *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*, BELIV '16, pages 125–132, New York, NY, USA, 2016. ACM.

[9] S. M. Borodo, S. M. Shamsuddin, and S. Hasan. Big data platforms and techniques. *Indonesian Journal of Electrical Engineering and Computer Science*, 1(1):191–200, 2016.

[10] C. Brunsdon, J. Corcoran, and G. Higgs. Visualising space and time in crime patterns: A comparison of methods. *Computers, Environment and Urban Systems*, 31(1):52–75, 2007.

[11] T. Butkiewicz, R. Chang, Z. Wartell, and W. Ribarsky. Visual analysis and semantic exploration of urban lidar change detection. In *Proceedings of the 10th Joint Eurographics / IEEE - VGTC Conference on Visualization*, EuroVis'08, pages 903–910, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[12] T. Butkiewicz, W. Dou, Z. Wartell, W. Ribarsky, and R. Chang. Multi-focused geospatial analysis using probes. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1165–1172, 2008.

[13] T. Butkiewicz, R. K. Meentemeyer, D. A. Shoemaker, R. Chang, Z. Wartell, and W. Ribarsky. Alleviating the modifiable areal unit problem within probe-based geospatial analyses. In *Computer Graphics Forum*, volume 29, pages 923–932. Wiley Online Library, 2010.

[14] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, OSDI '06, pages 15–15, Berkeley, CA, USA, 2006. USENIX Association.

[15] R. Chang, T. Butkiewicz, C. Ziemkiewicz, Z. Wartell, N. S. Pollard, and W. Ribarsky. Legible simplification of textured urban models. *IEEE Computer Graphics and Applications*, 28(3):27–36, 2008.

[16] R. Chang, M. Ghoniem, R. Kosara, W. Ribarsky, J. Yang, E. Suma, C. Ziemkiewicz, D. Kern, and A. Sudjianto. Wirevis: Visualization of categorical, time-varying data from financial transactions. In *2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 155–162, Oct 2007.

[17] O. Chapelle, P. Haffner, and V. N. Vapnik. Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, 10(5):1055–1064, 1999.

[18] C. P. Chen and C.-Y. Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314 – 347, 2014.

[19] M. Chen, S. J. Mountford, and A. Sellen. A study in interactive 3-d rotation using 2-d control devices. *SIGGRAPH Comput. Graph.*, 22(4):121–129, June 1988.

[20] L. Chittaro. Visualizing information on mobile devices. *Computer*, 39(3):40–45, March 2006.

[21] I. Cho, W. Dou, D. X. Wang, E. Sauda, and W. Ribarsky. Vairoma: A visual analytics system for making sense of places, times, and events in roman history. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):210–219, 2016.

[22] M. CHORZEPA, A. SAEIDPOUR, J. CHRISTIAN, and S. DURHAM. Hurricane vulnerability of coastal bridges using multiple environmental parameters. *International Journal of Safety and Security Engineering*, 6(1):10–18, 2016.

[23] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1237–1242. AAAI Press, 2011.

[24] C. Clasen, F. A. Kruse, and A. Kim. Analysis of lidar data for emergency management and disaster response. In *Imaging and Applied Optics Technical Papers*, page RTu2E.2. Optical Society of America, 2012.

[25] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew. Deep learning with cots hpc systems. In S. Dasgupta and D. Mcallester, editors, *Proceedings of the*

30th International Conference on Machine Learning (ICML-13), volume 28, pages 1337–1345. JMLR Workshop and Conference Proceedings, May 2013.

[26] 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.

[27] P. Craig. Interactive animated mobile information visualisation. In *SIGGRAPH Asia 2015 Mobile Graphics and Interactive Applications*, SA '15, pages 24:1–24:6, New York, NY, USA, 2015. ACM.

[28] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[29] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6):205–220, Oct. 2007.

[30] E. Delmelle, C. Dony, I. Casas, M. Jia, and W. Tang. Visualizing the impact of space-time uncertainties on dengue fever patterns. *International Journal of Geographical Information Science*, 28(5):1107–1127, 2014.

[31] E. Delmelle, C. Kim, N. Xiao, and W. Chen. Methods for space-time analysis and modeling: An overview. *International Journal of Applied Geospatial Research (IJAGR)*, 4(4):1–18, 2013.

[32] E. M. Delmelle, H. Zhu, W. Tang, and I. Casas. A web-based geospatial toolkit for the monitoring of dengue fever. *Applied Geography*, 52:144–152, 2014.

[33] U. Demšar and K. Virrantaus. Space–time density of trajectories: exploring spatio-temporal patterns in movement data. *International Journal of Geographical Information Science*, 24(10):1527–1542, 2010.

[34] P. Deutsch and J.-L. Gailly. Zlib compressed data format specification version 3.3. Technical report, RFC 1950, May, 1996.

[35] D. J. Dewitt, S. Ghandeharizadeh, D. A. Schneider, A. Bricker, H. I. Hsiao, and R. Rasmussen. The gamma database machine project. *IEEE Trans. on Knowl. and Data Eng.*, 2(1):44–62, Mar. 1990.

[36] V. DIACONITA. Approaches for parallel data loading and data querying. *Database Systems Journal BOARD*, page 78.

[37] J. Dietrich, C. Trahan, M. Howard, J. Fleming, R. Weaver, S. Tanaka, L. Yu, R. L. Jr., C.N, J. Dawson, Westerink, G. Wells, A. Lu, K. Vega, A. Kubach, K. Dresback, R. Kolar, C. Kaiser, and R. Twilley. Surface trajectories of oil transport along the northern coastline of the gulf of mexico. In *Continental Shelf Research*, 2012.

[38] S. M. Drucker, D. Fisher, R. Sadana, J. Herron, and m. schraefel. Touchviz: A case study comparing two interfaces for data analytics on tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2301–2310, New York, NY, USA, 2013. ACM.

[39] J. Duda. Asymmetric numeral systems as close to capacity low state entropy coders. *CoRR*, abs/1311.2540, 2013.

[40] T. Eaglin, I. Cho, and W. Ribarsky. Space-time kernel density estimation for real-time interactive visual analytics. In *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.

[41] T. Eaglin, K. Subramanian, and J. Payton. 3d modeling by the masses: A mobile app for modeling buildings. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 315–317, March 2013.

[42] T. Eaglin, X. Wang, and W. Ribarsky. Interactive visual analytics in support of image-encoded lidar analysis. *Electronic Imaging*, 2016(1):1–9, 2016.

[43] T. Eaglin, X. Wang, W. Ribarsky, and W. Tolone. Ensemble visual analysis architecture with high mobility for large-scale critical infrastructure simulations. In *SPIE/IS&T Electronic Imaging*, pages 939706–939706. International Society for Optics and Photonics, 2015.

[44] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9:66–104, 1990.

[45] O. ElTayeby, T. Eaglin, M. Abdullah, D. Burlinson, W. Dou, and L. Yao. Detecting drinking-related contents on social media by classifying heterogeneious data types. In *Proceedings of the 2017 International Conference on Industrial, Engineering, Other Applications of Applied Intelligent Systems*, IEA/AIE '17, 2017.

[46] R. Engs, D. Hanson, and B. Diebold. The drinking patterns and problems of a national sample of college students, 1994. 1996.

[47] D. Escobar, M. Prato, J. Bustos-Jimenez, and A. Lucero. Mobile information visualization design of the adkintun mobile app. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, UbiComp '16, pages 641–646, New York, NY, USA, 2016. ACM.

[48] J. Estornell, L. A. Ruiz, B. Velzquez-Mart, and T. Hermosilla. Analysis of the factors affecting lidar dtm accuracy in a steep shrub area. *Int. J. Digital Earth*, 4(6):521–538, 2011.

[49] J. Fan, F. Han, and H. Liu. Challenges of big data analysis. *National Science Review*, 1(2):293–314, 2014.

[50] A. Fuchs. Client-Side Data Transformation in Lustre. Master's thesis, Universitt Hamburg, 05 2016.

[51] P. S. Games and A. Joshi. An evaluation-guided approach for effective data visualization on tablets, 2015.

[52] S. Gao. Spatio-temporal analytics for exploring human mobility patterns and urban dynamics in the mobile age. *Spatial Cognition & Computation*, 15(2):86–114, 2015.

[53] S. Gao, Y. Hu, K. Janowicz, and G. McKenzie. A spatiotemporal scientometrics framework for exploring the citation impact of publications and scientists. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 204–213. ACM, 2013.

[54] J. Guest, T. Eaglin, K. Subramanian, and W. Ribarsky. Visual analysis of situationally aware building evacuations. volume 8654, pages 86540G–86540G–14, 2013.

[55] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh. Wtf: The who to follow service at twitter. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 505–514, New York, NY, USA, 2013. ACM.

[56] M. M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, Oct. 2008.

[57] E. J. Hastings, J. Mesit, and R. K. Guha. Optimization of large-scale, real-time simulations by spatial hashing. In *Proc. 2005 Summer Computer Simulation Conference*, volume 37, pages 9–17, 2005.

[58] T. Hermosilla, L. A. Ruiz, J. A. Recio, and J. Estornell. Evaluation of automatic building detection approaches combining high resolution images and lidar data. *Remote Sensing*, 3(6):1188–1210, 2011.

[59] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu. Starfish: A self-tuning system for big data analytics. In *CIDR*, volume 11, pages 261–272, 2011.

[60] A. Hohl, E. M. Delmelle, and W. Tang. Spatiotemporal Domain Decomposition for Massive Parallel Computation of Space-Time Kernel Density. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 7–11, July 2015.

[61] G. Horne. Beyond point estimates: Operational synthesis and data farming. *Maneuver Warfare Science*, 2001.

[62] X. Hu and H. Liu. Text analytics in social media. In *Mining text data*, pages 385–414. Springer, 2012.

[63] C. Ji, Y. Li, W. Qiu, U. Awada, and K. Li. Big data processing in cloud computing environments. In *2012 12th International Symposium on Pervasive Systems, Algorithms and Networks*, pages 17–23. IEEE, 2012.

[64] X. Jin, B. W. Wah, X. Cheng, and Y. Wang. Significance and challenges of big data research. *Big Data Research*, 2(2):59 – 64, 2015. Visions on Big Data.

[65] M. John, T. Ikuta, and J. Ferbinteanu. Graph analysis of structural brain networks in alzheimers disease: beyond small world properties. *Brain Structure and Function*, pages 1–20, 2016.

[66] S. Kaisler, F. Armour, J. A. Espinosa, and W. Money. Big data: Issues and challenges moving forward. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 995–1004, Jan 2013.

[67] M. Kallmann and M. Kapadia. Navigation meshes and real-time dynamic planning for virtual worlds. In *ACM SIGGRAPH 2014 Courses*, SIGGRAPH '14, pages 3:1–3:81, New York, NY, USA, 2014. ACM.

[68] T. Kapler and W. Wright. Geotime information visualization. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 25–32, Oct 2004.

[69] A. Karpathy and F. Li. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014.

[70] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1725–1732, Washington, DC, USA, 2014. IEEE Computer Society.

[71] E. Kaya, M. T. Eren, C. Doger, and S. Balcisoy. Do 3d visualizations fail? an empirical discussion on 2d and 3d representations of the spatio-temporal data.

[72] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Information visualization. chapter Visual Analytics: Definition, Process, and Challenges, pages 154–175. Springer-Verlag, Berlin, Heidelberg, 2008.

[73] E. D. Keim, J. Kohlhammer, and G. Ellis. Mastering the information age: Solving problems with visual analytics, eurographics association, 2010.

[74] J. Kim and C. Scott. Variable kernel density estimation. *Annals of Statistics*, 20:1236–1265, 1992.

[75] S. D. Kim, S. M. Lee, S. M. Lee, J. H. Jang, J.-G. Son, Y. H. Kim, and S. E. Lee. Compression accelerator for hadoop appliance. In *International Conference on Internet of Vehicles*, pages 416–423. Springer, 2014.

[76] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[77] D. Klein, J. Smarr, H. Nguyen, and C. D. Manning. Named entity recognition with character-level models. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 180–183, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[78] P. Klemm, K. Lawonn, S. Glaßer, U. Niemann, K. Hegenscheid, H. Volzke, and B. Preim. 3d regression heat map analysis of population study data. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):81–90, 2016.

[79] S. Ko, J. Zhao, J. Xia, S. Afzal, X. Wang, G. Abram, N. Elmqvist, L. Kne, D. Van Riper, K. Gaither, et al. Vasa: Interactive computational steering of large asynchronous simulation pipelines for societal infrastructure. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):1853–1862, 2014.

[80] M. J. Kraak. The space-time cube revisited from a geovisualization perspective. *Proceedings of the 21st International Cartographic Conference*, 1995, 1988.

[81] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[82] M. Krone, C. Müller, and T. Ertl. Remote rendering and user interaction on mobile devices for scientific visualization. In *Proceedings of the 8th International Symposium on Visual Information Communication and Interaction*, pages 21–26. ACM, 2015.

[83] M.-P. Kwan and D. M. Ransberger. Lidar assisted emergency response: Detection of transport network obstructions caused by major disasters. *Computers, Environment and Urban Systems*, 34(3):179 – 188, 2010.

[84] F. Lamberti and A. Sanna. A streaming-based solution for remote visualization of 3d graphics on mobile devices. *IEEE transactions on visualization and computer graphics*, 13(2), 2007.

[85] M. B. Lazreg, M. Goodwin, and O.-C. Granmo. Deep learning for social media analysis in crises situations. In *The 29th Annual Workshop of the Swedish Artificial Intelligence Society (SAIS) 2–3 June 2016, Malmö, Sweden*, page 31, 2016.

[86] B. Lee, P. Isenberg, N. H. Riche, and S. Carpendale. Beyond mouse and keyboard: Expanding design considerations for information visualization interactions. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2689–2698, 2012.

[87] T. Li, C. Luo, H. Chen, and J. Zhang. Pickt: A solution for big data analysis. In *International Conference on Rough Sets and Knowledge Technology*, pages 15–25. Springer, 2015.

[88] A. Liaw and M. Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[89] J. Liu, K. Lyons, K. Subramanian, and W. Ribarsky. Semi-automated processing and routing within indoor structures for emergency response applications. volume 7709, pages 77090Z–77090Z–10, 2010.

[90] Z. Liu, B. Jiang, and J. Heer. immens: Real-time visual querying of big data. In *Computer Graphics Forum*, volume 32, pages 421–430. Wiley Online Library, 2013.

[91] D. Lopez, L. Oehlberg, C. Doger, and T. Isenberg. Towards an understanding of mobile touch navigation in a stereoscopic viewing environment for 3d data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 22(5):1616–1629, May 2016.

[92] U. Lopez-Novoa, J. Sáenz, A. Mendiburu, and J. Miguel-Alonso. An efficient implementation of kernel density estimation for multi-core and many-core architectures. *International Journal of High Performance Computing Applications*, 29(3):331–347, 2015.

[93] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004.

[94] K. Lu, A. Chaudhuri, T.-Y. Lee, H.-W. Shen, and P. C. Wong. Exploring vector fields with distribution-based streamline analysis. In *Visualization Symposium (PacificVis), 2013 IEEE Pacific*, pages 257–264, Feb 2013.

[95] J. Lukasczyk, R. Maciejewski, C. Garth, and H. Hagen. Understanding hotspots: a topological visual analytics approach. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 36. ACM, 2015.

[96] R. Maciejewski, S. Rudolph, R. Hafen, A. Abusalah, M. Yakout, M. Ouzzani, W. S. Cleveland, S. J. Grannis, and D. S. Ebert. A visual analytics approach to understanding spatiotemporal hotspots. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):205–220, 2010.

[97] R. Mahanty and P. K. Mahanti. Unleashing artificial intelligence onto big data: A review. In *Handbook of Research on Computational Intelligence Applications in Bioinformatics*, pages 1–16. IGI Global, 2016.

[98] H. Mantz, K. Jacobs, and K. Mecke. Utilizing minkowski functionals for image analysis: a marching square algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(12):P12015, 2008.

[99] H. Mantz, K. Jacobs, and K. Mecke. Utilizing minkowski functionals for image analysis: a marching square algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(12):P12015, 2008.

[100] W. Medjroubi, C. Matke, and D. Kleinhans. Scigrid–an open source reference model for the european transmission network. 2015.

[101] S. Mittelstdt, X. Wang, T. Eaglin, D. Thom, D. Keim, W. Tolone, and W. Ribarsky. An integrated in-situ approach to impacts from natural disasters on critical infrastructures. In *System Sciences (HICSS), 2015 48th Hawaii International Conference on*, pages 1118–1127, Jan 2015.

[102] M. H. Murray. Storm-tide elevations produced by hurricane andrew along the southern florida coasts, august 24, 1992. Technical report, US Geological Survey, 1994.

[103] T. Nakaya and K. Yano. Visualising crime clusters in a space-time cube: An exploratory data-analysis approach using space-time kernel density estimation and scan statistics. *Transactions in GIS*, 14(3):223–239, 2010.

[104] Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[105] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. *CoRR*, abs/1503.08909, 2015.

[106] L. Nowell, E. Hetzler, and T. Tanasse. Change blindness in information visualization: a case study. In *IEEE Symposium on Information Visualization, 2001. INFOVIS 2001.*, pages 15–22, 2001.

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

[108] N. R. Pal and S. K. Pal. A review on image segmentation techniques. *Pattern recognition*, 26(9):1277–1294, 1993.

[109] V. Pascucci, D. E. Laney, R. Frank, G. Scorzelli, L. Linsen, B. Hamann, and F. Gygi. Real-time monitoring of large scientific simulations. In *Proceedings of the 18-th annual ACM Symposium on Applied Computing*, pages 194–198, Melbourne, Florida, March 2003.

[110] R. Raina, A. Madhavan, and A. Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 873–880, New York, NY, USA, 2009. ACM.

[111] A. C. Rice and O. J. Woodman. Crowd-sourcing world models with openroommap. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 764–767, March 2010.

[112] R. Richter and J. Dllner. Concepts and techniques for integration, analysis and visualization of massive 3d point clouds. *Computers, Environment and Urban Systems*, 45(0):114 – 124, 2014.

[113] A. Ritter, S. Clark, Mausam, and O. Etzioni. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1524–1534, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[114] J. Rivera, J. Leimhofer, and H.-A. Jacobsen. Opengridmap: towards automatic power grid simulation model generation from crowdsourced data. *Computer Science - Research and Development*, pages 1–11, 2016.

[115] J. C. Roberts. State of the art: Coordinated and multiple views in exploratory visualization. In *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 61–71, 2007.

[116] G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins. Animated visualization of multiple intersecting hierarchies. *Information Visualization*, 1(1):50–65, 2002.

[117] J. B. Rothnie, Jr., P. A. Bernstein, S. Fox, N. Goodman, M. Hammer, T. A. Landers, C. Reeve, D. W. Shipman, and E. Wong. Introduction to a system for distributed databases (sdd-1). *ACM Trans. Database Syst.*, 5(1):1–17, Mar. 1980.

[118] J. M. Rzeszotarski and A. Kittur. Kinetica: Naturalistic multi-touch data visualization. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 897–906, New York, NY, USA, 2014. ACM.

[119] R. Sadana and J. Stasko. Designing and implementing an interactive scatterplot visualization for a tablet computer. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, AVI '14, pages 265–272, New York, NY, USA, 2014. ACM.

[120] R. Sadana and J. Stasko. Designing multiple coordinated visualizations for tablets. In *Computer Graphics Forum*, volume 35, pages 261–270. Wiley Online Library, 2016.

[121] R. Sadana and J. Stasko. Expanding selection for information visualization systems on tablet devices. In *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces*, ISS '16, pages 149–158, New York, NY, USA, 2016. ACM.

[122] E. Santos, J. Freire, C. Silva, A. Khan, J. Tierny, B. Grimm, L. Lins, V. Pascucci, S. A. Klasky", R. D. Barreto, and N. Podhorszki. Enabling advanced visualization tools in a simulation monitoring system. In *Proceedings of the 5th IEEE International Conference on e-Science*, pages 358–365. IEEE, December 2009.

[123] R. Scheepens, N. Willems, H. van de Wetering, G. Andrienko, N. Andrienko, and J. van Wijk. Composite density maps for multivariate trajectories. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2518–2527, Dec 2011.

[124] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[125] M. F. Schwartz and D. Wood. Discovering shared interests using graph analysis. *Communications of the ACM*, 36(8):78–89, 1993.

[126] M. Sedlmair, M. Meyer, and T. Munzner. Design Study Methodology: Reflections from the Trenches and the Stacks. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)*, 18(12):2431–2440, 2012.

[127] H. Sencan, Z. Chen, W. Hendrix, T. Pansombut, F. Semazzi, A. Choudhary, V. Kumar, A. V. Melechko, and N. F. Samatova. Classification of emerging extreme event tracks in multivariate spatio-temporal physical systems using dynamic network structures: application to hurricane track prediction. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1478. Citeseer, 2011.

[128] S. Shekhar, K. Yang, V. M. Gunturi, L. Manikonda, D. Oliver, X. Zhou, B. George, S. Kim, J. M. Wolff, and Q. Lu. Experiences with evacuation route planning algorithms. *International Journal of Geographical Information Science*, 26(12):2253–2265, 2012.

[129] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[130] S. Shi, K. Nahrstedt, and R. Campbell. A real-time remote rendering system for interactive mobile graphics. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 8(3s):46, 2012.

[131] B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[132] A. Slingsby, J. Dykes, and J. Wood. Exploring uncertainty in geodemographics with interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2545–2554, Dec 2011.

[133] E. Smistad, A. C. Elster, and F. Lindseth. Real-time surface extraction and visualization of medical images using opencl and gpus. *Norsk informatikkonferanse*, pages 141–152, 2012.

[134] Y. Song, S. Pan, S. Liu, F. Wei, M. X. Zhou, and W. Qian. Constrained text coclustering with supervised and unsupervised constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1227–1239, 2013.

[135] G.-D. Sun, Y.-C. Wu, R.-H. Liang, and S.-X. Liu. A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. *Journal of Computer Science and Technology*, 28(5):852–867, 2013.

[136] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[137] D. Thompson, J. Levine, J. Bennett, P.-T. Bremer, A. Gyulassy, V. Pascucci, and P. Pebay. Analysis of large-scale scalar data using hixels. In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pages 23–30, Oct 2011.

[138] C. Toth and G. Jóźków. Remote sensing platforms and sensors: A survey. *ISPRS Journal of Photogrammetry and Remote Sensing*, 115:22–36, 2016.

[139] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 5:1–5:16, New York, NY, USA, 2013. ACM.

[140] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(6):583–598, 1991.

[141] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[142] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014.

[143] J. Walker, R. Borgo, and M. W. Jones. Timenotes: a study on effective chart visualization and interaction techniques for time-series data. *IEEE transactions on visualization and computer graphics*, 22(1):549–558, 2016.

[144] L. Wang, Y. Wang, C. Yang, and J. D. Owens. A comparative study on exact triangle counting algorithms on the gpu. In *Proceedings of the ACM Workshop on High Performance Graph Processing*, HPGP '16, pages 1–8, New York, NY, USA, 2016. ACM.

[145] S. Wang and M. P. Armstrong. A quadtree approach to domain decomposition for spatial interpolation in grid computing environments. *Parallel Computing*, 29(10):1481–1504, 2003.

[146] X. Wang, W. Dou, T. Butkiewicz, E. A. Bier, and W. Ribarsky. A two-stage framework for designing visual analytics system in organizational environments. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 251–260. IEEE, 2011.

[147] Y. Wang, A. Davidson, Y. Pan, Y. Wu, A. Riffel, and J. D. Owens. Gunrock: A high-performance graph processing library on the gpu. In *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '16, pages 11:1–11:12, New York, NY, USA, 2016. ACM.

[148] D. Wigdor and D. Wixon. *Brave NUI world: designing natural user interfaces for touch and gesture*. Elsevier, 2011.

[149] J. Wood, D. Badawood, J. Dykes, and A. Slingsby. Ballotmaps: Detecting name bias in alphabetically ordered ballot papers. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2384–2391, Dec 2011.

[150] J. Wood and J. Dykes. Spatially ordered treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1348–1355, Nov 2008.

[151] Y. Wu, Y. Wang, Y. Pan, C. Yang, and J. D. Owens. Performance characterization of high-level programming models for gpu graph analytics. In *Workload Characterization (IISWC), 2015 IEEE International Symposium on*, pages 66–75, Oct 2015.

[152] A. Zerger and D. I. Smith. Impediments to using {GIS} for real-time disaster decision support. *Computers, Environment and Urban Systems*, 27(2):123 – 141, 2003.

[153] H. Zhang, G. Chen, B. C. Ooi, K.-L. Tan, and M. Zhang. In-memory big data management and processing: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 27(7):1920–1948, 2015.

[154] J. Zhang, Y. Song, C. Zhang, and S. Liu. Evolutionary hierarchical dirichlet processes for multiple correlated time-varying corpora. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 1079–1088, New York, NY, USA, 2010. ACM.

[155] Y. Zhao, J. Wu, and C. Liu. Dache: A data aware caching for big-data applications using the mapreduce framework. *Tsinghua Science and Technology*, 19(1):39–50, Feb 2014.

[156] Y. Zhou, T. H. D. Dao, J.-C. Thill, and E. Delmelle. Enhanced 3d visualization techniques in support of indoor location planning. *Computers, Environment and Urban Systems*, 50:15–29, 2015.

[157] A. Zielinski, S. E. Middleton, L. N. Tokarchuk, and X. Wang. Social media text mining and network analysis for decision support in natural crisis management. *Proc. ISCRAM. Baden-Baden, Germany*, pages 840–845, 2013.

APPENDIX A: IRB INFORMED CONSENT FORM

UNCCHARLOTTE
The University of North Carolina at Charlotte

Informed Consent for
Evaluating Mobile Visual Analytics for Hurricane Simulations

Project Purpose

In this study we will be evaluating a tablet based visual analytics application on hurricane simulations using two group of graduate students from the visualization center and the geography department.

Investigators

Isaac Cho, Computer Science
Todd Eaglin, Computer Science

Eligibility

You may participate in this study if you are above 18 and if you can comfortably use your arms and fingers and communicate in spoken English.

Overall Description of Participation

The experiment will adopt a between subject design. You will be provided with a tablet and software application. All surveys and interviews will be done on paper as written responses. The interview will occur at the end. The interview will be a set of open-ended questions to which you will respond in writing.

Study procedure:

1. We will ask you to read and sign the consent form if you wish to participate.

2. You will fill out a questionnaire on basic user and demographic information. This step will take about 3 minutes.

3. We will provide you 10 minutes of training on how to use the tablet visual interface

4. Task 1 (2D view + Draw a hurricane path + initial analysis):
You will be asked to draw a hypothetical hurricane path similar to that of hurricane Matthew. Participants will be shown an image of hurricane Matthew's path as a reference and instructed on how to draw a path on the tablet device.

You will then be asked to analyze the temporal results from the path they just drew. After that they will be asked to spatially analyze the path they drew.

5. Task 2 (3D View + selection):
You will then be asked to use the 3D view, which incorporates the space-time kernel density estimation analytical technique and visualization. We will explain beforehand to you what the technique is how it works and what the visualization means. We will also ask you to use

volumetric rendering visualization techniques and interact directly with the Space-Time Kernel Density Estimation through selection.

6. Task 3 (Temporal/Spatial Slicing):
In this section we will ask you to perform two different types of slicing techniques. The first is temporal slicing and the second is spatial slicing. As part the temporal slicing. We will then ask you to select a region in time and then visually connect that to the space-time kernel density estimation 3D structure. We will then further ask you to analyze the temporal slice and report the disabled infrastructure within the slice.

For spatial slicing we will ask participants to create a slicing plane to cut the space-time kernel density estimation 3D structure on the spatial axis producing a unique analytical result. We will then ask you to further analyze the spatial slice and report the disabled infrastructure within the slice.

7. You will be asked to fill out a post study questionnaire regarding the ease of use and feedback on the user interface. This step will take about 5 minutes.

Length of Participation

Participation should take approximately 50-70 minutes.

Risks and Benefits of Participation
Volunteer Statement

You are a volunteer. The decision to participate in this study is completely up to you. If you decide to participate in the study, you may stop at any time. You will not be treated any differently if you decide not to participate in the study or if you stop once you have started. You can request to withdraw your segment after the testing is complete.

Confidentiality Statement

Any information about your participation, including your identity, is completely confidential. The following efforts will be taken to protect confidentiality and privacy:

1) The informed consent form will be kept in a locked filing cabinet, separate from the rest of the data.
2) You will be assigned a random ID consisting two randomly-generated initials (initials will not correspont to participants' name). The participants will only be referred by assigned alphanumeric codes both in internal communication between researchers or in the form of written reports.
3) All recorded files and data during the study will be kept in the Charlotte Visualization Center (room 437 in Woodward Hall) on password-protected computers and in locked cabinets. The files will be destroyed after two years by investigators under the guidance of the responsible faculty.

Statement of Fair Treatment and Respect

UNC Charlotte wants to make sure that you are treated in a fair and respectful manner. Contact the university's Research Compliance Office (704-687-1871) if you have questions about how you are treated as a study participant. If you have any questions about the actual project or study, please contact Dr. Isaac Cho (icho1@uncc.edu), or Todd Eaglin(teaglin@uncc.edu).

Participant Consent

I have read the information in this consent form. I have had the chance to ask questions about this study, and those questions have been answered to my satisfaction. I am at least 18 years of age, and I agree to participate in this research project. I understand that I will receive a copy of this form after it has been signed by me and the principal investigator of this research study.

Participant Name (PRINT)	DATE
Participant Signature	
Investigator Signature	DATE

APPENDIX B: IRB DEMOGRAPHICS QUESTIONS

1. Your given ID number (Instructor only):

2. Your age:

3. Your gender:

4. Occupational Status: Undergraduate student ___

    Master Student ___

    PhD Student___

    Research Assistant/Fellow ___

    Staff-systems, technical ___

    Faculty ___

    Administrative Staff ___

    Other: __________________

5. Your major:

6. Are you colorblind? : Yes / No

7. Do you have any disabilities or injuries that might limit your ability to use either of your left or right arm, hand and/or fingers in everyday tasks such as writing, painting, using a computer mouse or advanced game controller?   Yes/ No

9. Are you familiar with multi touch input? Yes / No

APPENDIX C: IRB PRE-QUESTIONAIRE

Pre-Survey Questionnaire:

On a scale of 1-7 how familiar are you with using a tablet device:

1	2	3	4	5	6	7
Not at all			Somewhat		Very Familiar	

What tools do you use to visualize and understand space-time data?

What is the most important feature you use from those tools?

APPENDIX D: IRB POST-QUESTIONAIRE

Post- Survey Questionnaire:

Task 1 (2D view + Draw a hurricane path + initial analysis):
Participants will be asked to draw a hypothetical hurricane path similar to that of hurricane Matthew. Participants will be shown an image of hurricane Matthew's path as a reference and instructed on how to draw a path on the tablet device.

Participants will then be asked to analyze the temporal results from the path they just drew. After that they will be asked to spatially analyze the path they drew.

1) How difficult was it to draw a hurricane path on the tablet device?

   1 2 3 4 5 6 7

   Very Hard Moderate Very Easy

2) How difficult was it to find the peak time period of when outages occurred?

   1 2 3 4 5 6 7

   Very Hard Moderate Very Easy

3) How difficult was it to spatially locate the peak area of outages?

   1 2 3 4 5 6 7

   Very Hard Moderate Very Easy

4) How difficult was it to navigate in the 2D view?

   1 2 3 4 5 6 7

   Very Hard Moderate Very Easy

Task 2 (3D View + selection):
Participants will then be asked to use the 3D view, which incorporates the space-time kernel density estimation analytical technique and visualization. We will explain beforehand to participants what the technique is how it works and what the visualization means. We will also ask participant to use volumetric rendering visualization techniques and interact directly with the Space-Time Kernel Density Estimation through selection.

1) How difficult was it to navigate in the 3D view?

1 2 3 4 5 6 7

Very Hard Moderate Very Easy

2) How familiar are you with the Space-Time Kernel Density Estimation technique?

1 2 3 4 5 6 7

Not at all Somewhat Very Familiar

3) How difficult was it to identify spatial and temporal hotspots?

1 2 3 4 5 6 7

Very Hard Moderate Very Easy

4) How familiar are you with Volumetric Rendering?

1 2 3 4 5 6 7

Not at all Somewhat Very Familiar

5)How helpful is the addition of volumetric rendering?

1 2 3 4 5 6 7

Not at all Somewhat Very Helpful

6)How easy is it to select a Space-time Kernel Density Estimation structure?

1 2 3 4 5 6 7

Very Hard Moderate Very Easy

7) How insightful is being able to select a Space-time kernel density estimation structure and view the specific outages?

1 2 3 4 5 6 7

Not at all Somewhat Very

Task 3 (Temporal/Spatial Slicing):
In this section we ask participants to perform two different types of slicing techniques. The first is temporal slicing and the second is spatial slicing. As part the temporal slicing. We will ask participants to select a region in time and then visually connect that to the space-time kernel density estimation 3D structure. We will then further ask them to analyze the temporal slice and report the disabled infrastructure within the slice.

For spatial slicing we will ask participants to create a slicing plane to cut the space-time kernel density estimation 3D structure on the spatial axis producing a unique analytical result. We will then ask participants to further analyze the spatial slice and report the disabled infrastructure within the slice.

1) How insightful is temporal slicing to further analyze the space-time kernel density estimation?

1 2 3 4 5 6 7

Not at all Somewhat Very

2)How difficult was it to perform temporal slicing on a tablet device?

1 2 3 4 5 6 7

Very Hard Moderate Very Easy

3) How insightful is spatial slicing to further analyze the space-time kernel density estimation?

1 2 3 4 5 6 7

Not at all Somewhat Very

4) How difficult was it to perform spatial slicing on a tablet device?

1	2	3	4	5	6	7
Very Hard			Moderate			Very Easy

Open Ended Interview Questions:

1) How does this tool compare to any other tools that you've used?

2) Could you see other applications of this tool for mobile devices?

3) What improvements would you like to see added to this tool?