ABSTRACT

KHATRI, HILAY MANSUKHBHAI. Data Visualization using Web Technologies. (Under the direction of Dr. Christopher Healey.)

Visualization is an important area of research that allows viewers to apply their visual perception to identify characteristics in data represented as an image. Traditionally, visualization tools are created for desktop platforms but this can introduce accessibility, portability or distribution issues. Recently, it was proposed that visualization tools be deployed as web applications that can be accessed from web browsers. There are many issues when building a web based visualization such as which techniques to use for drawing the visualizations, how to provide interactivity and what strategies to use for data management. In order to develop a visualization tool, one can either use existing libraries or build the tool "from scratch". We selected, then extended, an existing library. We also explored two user interface libraries to allow a user to control the visualization. Finally, we evaluated two different approaches for managing data. We applied these techniques to develop visualization tools for the National Collaborative for Bio-Preparedness (NCBP) project and the Missing Meals of North Carolina (MMNC) project. Our visualizations run in any modern browser. We received positive feedback from domain experts in both the projects, suggesting our tools provide effective visualizations.

Data Visualization using Web Technologies

by
Hilay Mansukhbhai Khatri

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2013

APPROVED BY:

Dr. Benjamin Watson	Dr. Robert St. Amant

Dr. Christopher Healey
Chair of Advisory Committee

DEDICATION

To my family and friends.

BIOGRAPHY

Hilay Khatri graduated from L. J. Institute of Engineering and Technology, Ahmedabad, India with a Bachelors degree in Computer Engineering. He came to North Carolina State University to pursue Masters in Computer Science. He also interned at Red Hat during the summer of 2012. After completing his studies, he is planning to join IT industry as a Software Developer.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Christopher Healey for guiding and supporting my thesis. You are a great mentor, instructor, researcher and it was an honor working with you.

I would also like to thank Dr. Robert St. Amant and Dr. Benjamin Watson for taking their time off to review my thesis.

Next, I would like to thank Adam Marrs, John Slankas, Parikshit Deshpande, Prajakta Hegde, Dr. Laurie William, David Yoshikawa, Tariq Wilison, John Streck, Marc Hoit and all the members of the NCBP project.

I would also like to thank Dr. Aric LaBarr, Ray Holland, Charlie Hale and all the members of the MMNC project.

I am also thankful to my friends Srinath Ravindran, Kalpesh Padia, Lihua Hao and Prairie Goodwin for their support.

Lastly, I would like to thank my family for their constant support and motivation.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Over the last few years, the amount of data generated has increased exponentially in both size and complexity [1]. It is becoming more difficult to identify trends or patterns effectively. The field of visualization emerged from this need to analyze complex data with the hope that performing visual analysis on visual representations of the data would prove more effective than studying or analyzing the raw data directly. Visualizations are widely used in reporting, monitoring, surveys and scientific applications. Generally, visualizations are divided into two types:

1. Scientific Visualization: This area deals with realistic rendering of physical phenomena like meteor showers, light illumination, light rays scattering or shadows with a given spatial representation.

2. Information Visualization: This area deals with visualizing more abstract numeric, non-numeric, geographic, text or image data by choosing both a spatial and a visual representation.

1.1 Problem Description

Visualizing data has always been challenging. We need a good understanding of the analysis tasks to be performed on the data to be visualized and the tools required to build it. Generally, there are two ways to build a visualization tool: 1) Using an existing visualization toolkits or 2) Building from scratch. There are many issues in building a visualization tool including portability, cross-platform compatibility, display, user interface (UI), data management and accessibility. Traditionally, visualization tools are developed for desktop platforms but this can introduce issues with porting and distributing between different architectures. These issues can be resolved to some extent by deploying a visualization tool as a web based application, since they can be easily accessed from any web browser. The capability of such applications depends on the technology being used. For example, some visualizations require external plugins. This can lead to accessibility issues if the web browser is missing such plugins. If web technologies are used for building a visualization then there are questions regarding: 1) Rendering, 2) Interactive UI and 3) Data management.

There are many graphics packages available for desktop platforms to draw visualizations, including some that use hardware acceleration to provide optimized rendering. There are fewer technologies available for drawing objects on a web page. We chose to extend an existing visualization library for rendering our data. The resultant visualization is cross-browser compatible and does not depend on any external plugins.

Building an interactive UI for the visualization tool is also challenging but necessary since it establishes a communication channel between the user and the visualization. UI allows users to modify different properties of the visualization, control different layers of information, and zoom, scroll or drag the visualization to obtain a desired view. Various

libraries are available to help in building a web based UI. We studied the use of two existing libraries in our tools.

It is important to have a way to manage the data being visualized. Database management systems (DBMS) can be used to store, retrieve and process data. Adding a DBMS requires supporting a technology that can communicate with a database server to retrieve appropriate data. Another option is to read the data directly from a file. We used a DBMS to mange the data for the National Collaborative for Bio-Preparedness (NCBP) project. For the Missing Meals of North Carolina (MMNC) project we read data directly from a file.

1.2 Thesis Overview

In this thesis, we demonstrate the use of the web technologies to develop a 2D visualization. We began developing our tools by researching existing libraries and technologies capable of drawing a visualization on a web page. After studying various trade offs, we chose to visualize the data with an existing library. By using HTML's built-in interfaces and external user interface libraries, we provide interactive control over the visualization. We implemented both a DBMS based and a flat-filed based approach for handling data. We applied these strategies to two real world projects: 1) NCBP and 2) MMNC. Data is visualized on a 2D choropleth map. Figure 1.1 shows a visualization of the volume of medical alerts reported by county in the state of North Carolina. This visualization was developed with web technologies as part of the NCBP project. Here, the average number of alerts per day are shown using different shades of red and blue. Yellow circles highlight dense spatial clusters of alerts.

Our visualizations run in Firefox, Chrome, Safari, Opera and Internet Explorer. We

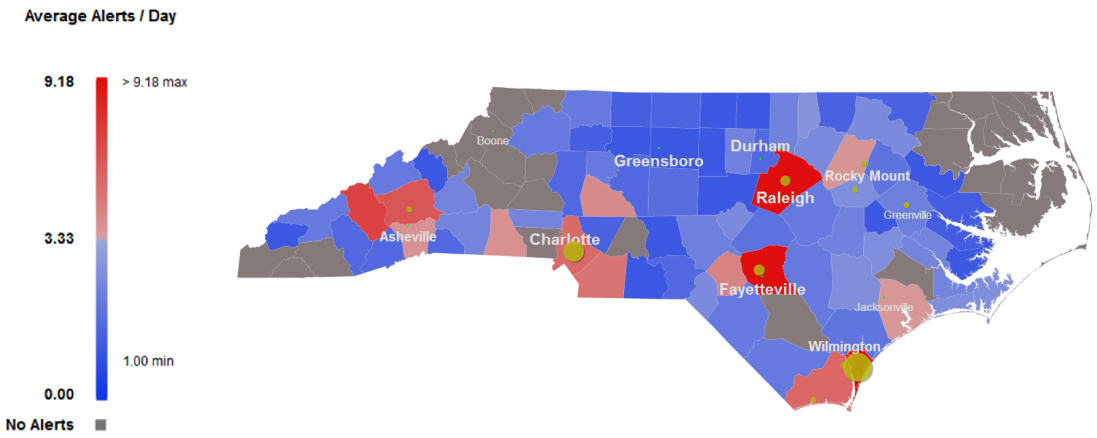

Figure 1.1: Visualizing medical alerts for the NCBP project on North Carolina state

verified the capabilities of the visualizations using anecdotal feedback obtained from subject experts. Results are positive, suggesting that web based visualizations are capable of producing effective analytical support.

Chapter 2

Background Work

There are many visualization tools developed over the years for researching and analyzing data. These tools are sometimes built for specific problem domains or data types. Some tools have a complicated development environment and can be difficult to learn. Moreover, tools are also architecture specific leading to accessibility and portability issues. These concerns led to the Visualization Toolkit (VTK), a visualization library designed to provide a basic programming platform to construct visualizations [34]. VTK is an aggregation of different components that can help in building a visualization pipeline by providing the ability to read, filter, and render data [34]. It supports a wide range of algorithms to visualize different types of data and can provide interaction through many of its widgets [64]. It is cross platform compatible and open source. However, it is a complex system that requires users to write programs. Various commercial visualization tools use VTK as a foundation. These tools are easier to use and extend since they do not require an in-depth understanding of VTK. With the popularity of web technologies, Javascript libraries have also emerged to support visualization on a web page.

2.1 Visualization Techniques

There are many techniques developed to visualize scalar, vector, text or image data. These techniques use charts, graphs or other advanced visualization algorithms and are employed in different fields like medical, engineering, navigation, education and social media.

2.1.1 Visualization by Data Type

Various visualization techniques have been developed for different types of data: scalar, vector, tensor, text or image.

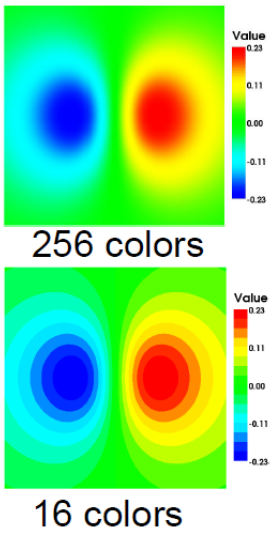

Figure 2.1: Scalar data mapped to different colors [68]

Scalar data: A scalar data is a 1D data attribute measuring magnitude. Probability, weight, height, etc., are some examples of scalar data. Color mappings are commonly used to visualize scalar data. Figure 2.1 shows scalar data represented with either 256

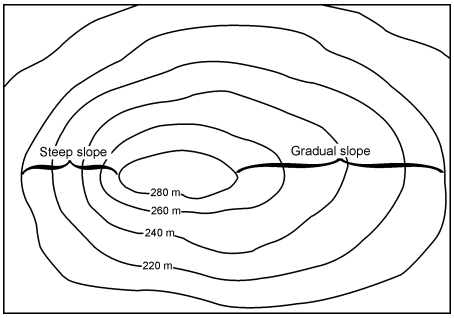

Figure 2.2: Contour visualization representing elevation [62]

colors or 16 colors using a "rainbow" color map. The mapping is implemented as a color lookup table with the scalar value discretized and used as an index into the color map. Sometimes, external information such as an allowable maximum and minimum of the scalar data are also considered in the lookup table. If the scalar value crosses these extreme, the mapping is clamped to either the minimum or maximum. Another technique uses contours to show equal value boundaries in the data. 2D contours can be seen in weather and topological maps to represent pressure and elevation boundaries. Figure 2.2 shows a contour map visualizing elevation.

Vector data: A vector is defined by its direction and magnitude. Vector data are often associated with fluid flow, velocity, or acceleration. There are many visualization techniques developed for visualizing vector data. A simple approach is to draw a line for each vector starting at the associated point, oriented and scaled based on the vector components [69]. Another technique called vector color coding works similar to color mapping for scalar data. Here, hue represents orientation and luminance represents magnitude, as seen in Figure 2.3 [69].

Tensors data: Tensors are geometrical entities that conceptualize scalars, vectors and linear operators independent of any co-ordinate system to express a linearized relation

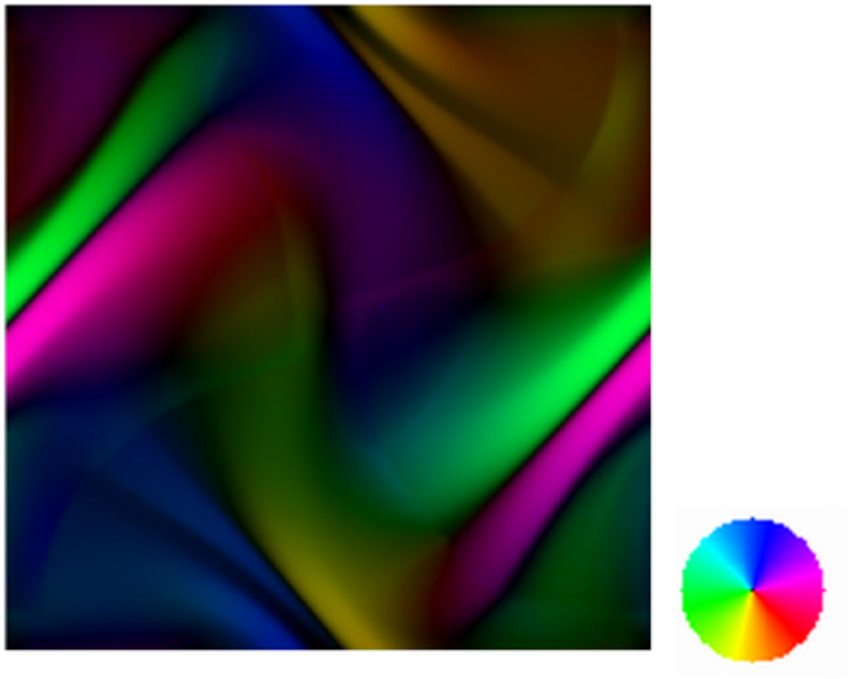

Figure 2.3: Vector visualization with hue representing orientation and luminance representing magnitude [69]

between multi-dimensional parameters [25]. Tensors are widely used in geomechanics, solid state physics and medical imaging. Visualizing tensor data is difficult because of their complexity. There are different techniques for visualizing tensor data but they are normally customized to a specific application or extended from vector field visualization [25]. Tensors can be visualized using icons, for example ellipsoids, where the principal axis is aligned to eigendirections and scaled according to eigenvalues [25]. Figure 2.4 shows an ellipsoid tensor visualization for a diseased spinal cord in a mouse with left-right asymmetries. Hyperstreamlines are also used to visualize tensor data. These provide more information than icon based methods but often produce cluttering [25]. Another approach considers tensor data as a force field that deforms objects. The deformation

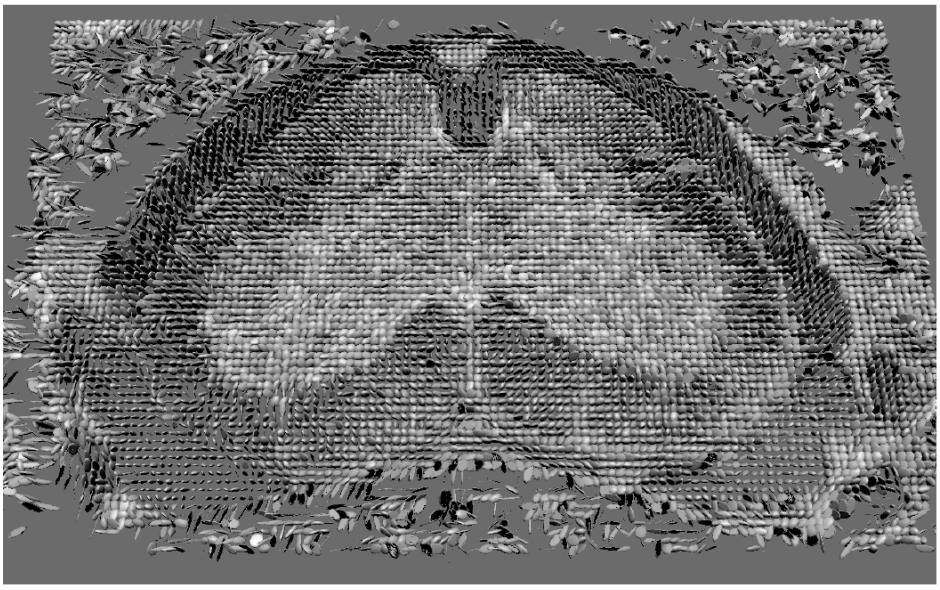

Figure 2.4: Ellipsoid representation for a diseased spinal cord in a mouse [32]

represents the tensor data [72].

Text data: Text visualization enables discovery of important information from textual content through graphical representation. One famous technique to visualize text is a tag cloud where the size of the text is rendered proportional to its frequency of occurrence in a document [8]. Figure 2.5 shows a tag cloud visualization of the text in this thesis chapter. Another approach for text visualization represents textual data in a vector space model. The dimensions of each vector are often high since numerous values can be derived for each text document. Dimension reduction techniques are usually applied to map these high dimensional vectors into 2D or 3D space [55]. Recently, text visualization has concentrated on the development of improved summarizing algorithms [52]. Many text visualization systems use keywords or phrases to summarize documents. Sammon was the first text visualization system built to automatically organize textual data [52]. SPIRE, developed by Pacific Northwest National laboratory can also create, search, and

Figure 2.5: Tag cloud for background work chapter of this thesis

interact with graphical representation of textual data [52].

Image data: Image capture has become commonplace due to proliferation of smartphones and point and shoot cameras. Images serve as a valuable source of information. Organizing and understanding images has become more critical as the size of publicly available common image collections has grown. One way to visualize images is to represent each image as a point set [17]. When a point is selected, the corresponding image can be viewed. Another visualization technique uses image similarity to place similar images closer to each other [36]. Figure 2.6 shows an image visualization system with the upper left corner representing images as points, the bottom left showing an enlarged image and the main screen showing representative sets.

Multivariate data: Sometimes, multiple attributes attached to each data element have to be visualized at the same time. The parallel co-ordinates technique uses vertical axes representing data attributes, with each attribute arrayed over the axis. A data element forms a polygonal line intersecting the axes at its corresponding value locations [41]. This technique helps in identifying correlation between attributes and differences in

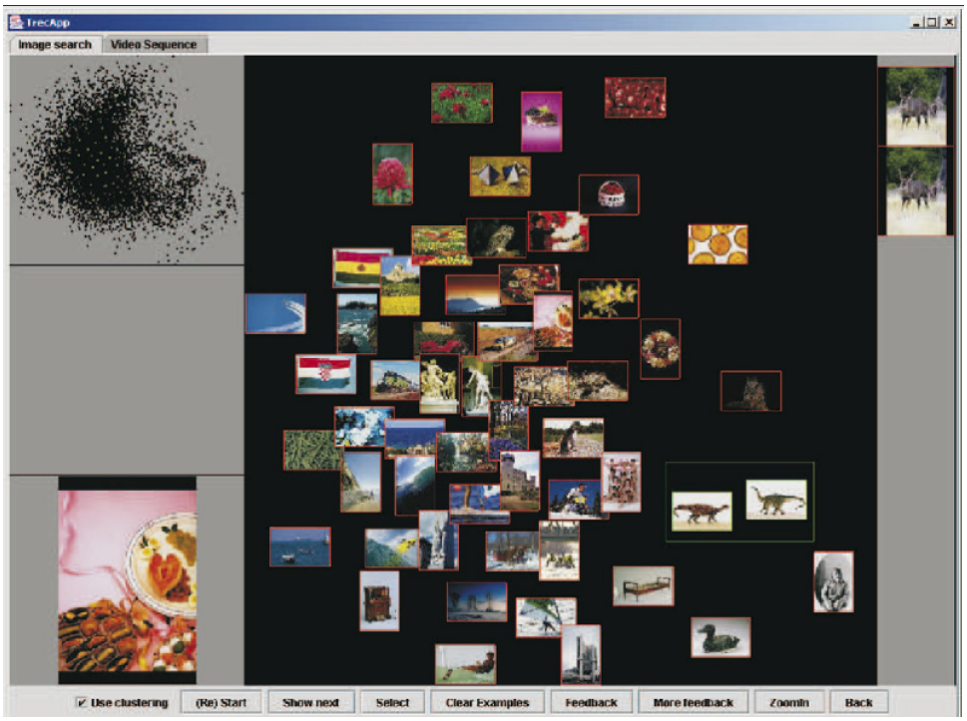

Figure 2.6: An image visualization system [36]

the data. It also helps to identify outliers [41]. However, for large number of data item, this technique can be confusing due to the high number of crossing lines [41]. Figure 2.7 shows a parallel co-ordinate visualization technique to visualize a 5D dataset. The radial co-ordinate visualization (RadViz) technique works similar to parallel co-ordinates but with axes emanating from the center of the circle. This technique is useful for finding clusters in a dataset [41, 38]. Glyphs are also used to visualize multivariate data. Glyphs are different icons or markers whose properties like shape, appearance or size are altered to visualize multiple attributes of the data [53]. Figure 2.8 shows a flower glyph visualizing different attributes of a web page [7]. Petals define the number of keywords, leaves define number of outgoing links, stem define document length and ground define number of incoming links.

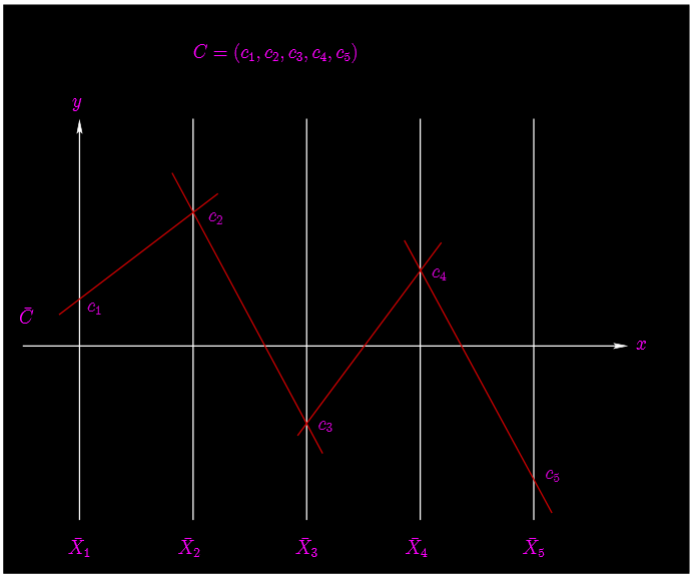

Figure 2.7: Parallel co-ordinates visualizing 5D data [26]

Figure 2.8: A flower glyph for visualizing multiple web page attributes [7]

2.1.2 Visualization by Domain

Visualizing data is useful in numerous domains such as engineering, medical, navigation, geographic assessment, education and research as it helps in analyzing data captured from different sources and applications.

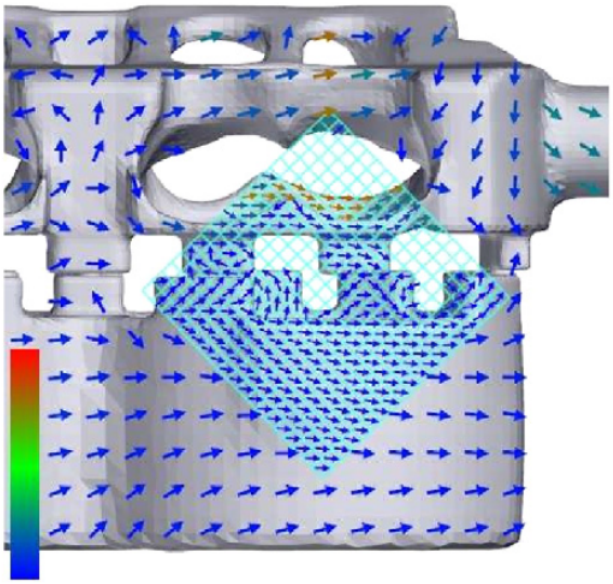

Figure 2.9: Flow visualization on the surface of cooling jacket [12]

Flow visualization deals with vector data and is widely used to study the behavior of liquids and gases. For example, it is used to study air flow around an engine to improve the cooling or to visualize blood flow for designing heart pumps [12]. Figure 2.9 shows the flow at the surface of a cooling jacket with color mapped to velocity.

Volume data is frequently used in medical applications [59]. Images obtained from MRI, ultrasound and other imaging devices are stacked together to form a 3D model [47]. Augmented reality tools and techniques can help doctors plan a surgery by visualizing

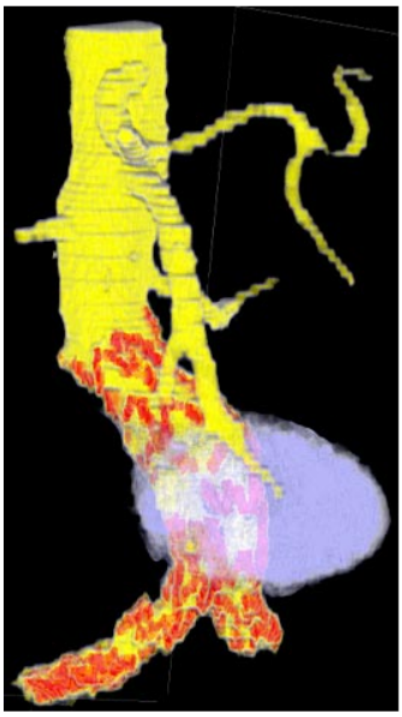

Figure 2.10: Volume rendering of Abdominal Aortic Aneurysms [59]

organs [47, 59]. Figure 2.10 shows a volume visualization used to help doctors during surgical planning of abdominal aortic aneurysms.

Terrain visualizations and Geographical Information Systems (GIS) are another common visualization domains. With the improvement of sensors and advancement in computational technology, it is possible to collect huge amounts of data to build terrain models. Terrain visualization is used to plan flood control systems, topographic surveys, environmental assessments and hydro power plants [71]. As shown in Figure 2.11, advanced terrain visualization techniques can be used to assist in navigation by deforming a map according to points of interests, thereby improving the display of route information [35].

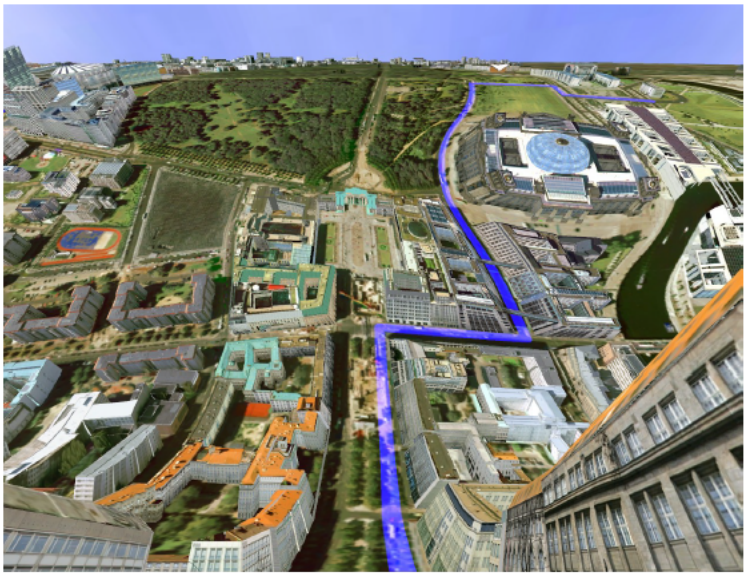

Figure 2.11: Terrain visualization for assisted navigation [35]

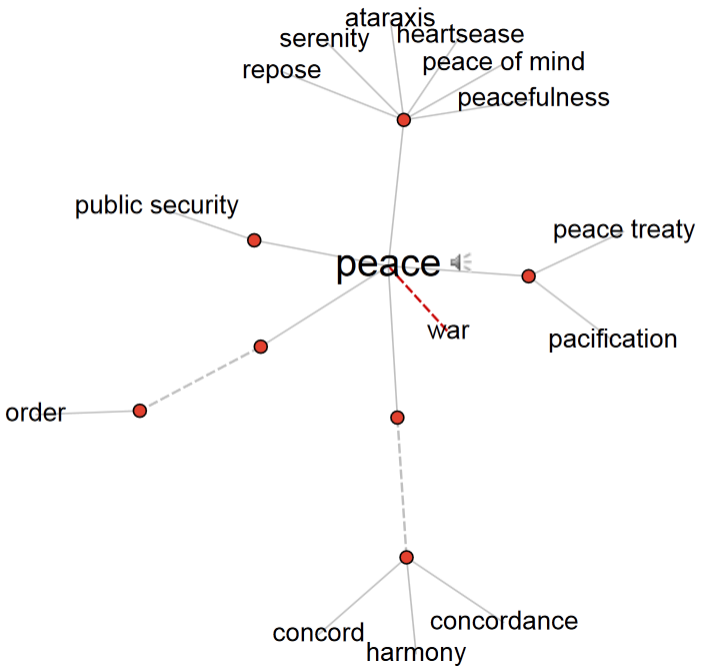

Figure 2.12: Visual thesaurus produced for peace keyword [60]

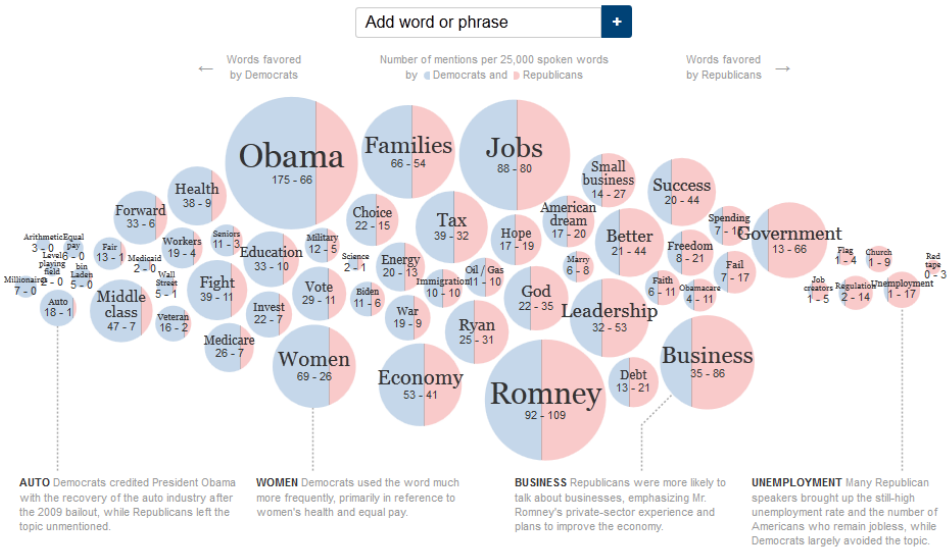

Figure 2.13: Visualization of commonly used words or phrases during presidential campaign [6]

Text visualizations are used to convey different types of information. As seen in Figure 2.12, Visual Thesaurus creates a word map and branches it to related words. It can help in improving vocabulary by visualizing how different words are connected [60]. During the presidential campaign, a visualization was used to compare the words or phrases used during the two presidential nominating conventions, as seen in Figure 2.13. Twitter has been a powerful source for studying people's opinions. Figure 2.14 shows a tweet visualization that pulls in tweets containing keywords, then estimates and visualizes the sentiment contained in each tweet [22].

Image visualization is also gaining in importance. There are many websites such as Flickr, Facebook and Panoramio that allow users to upload images. These images can serve as the raw data for a visualization. Figure 2.15 shows a visualization system that helps to identify geographic locations with high activity, thereby discovering potentially

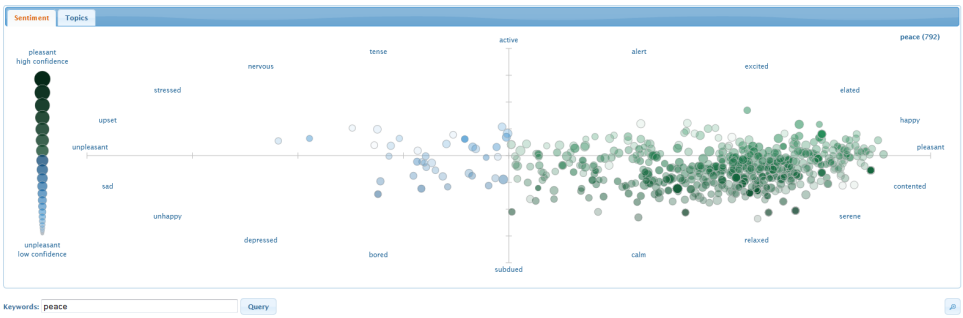

Figure 2.14: Tweet sentiment visualization [22]

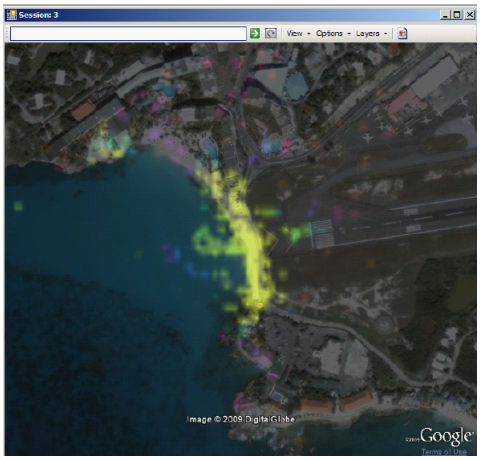

Figure 2.15: Visualization of attractive areas [30]

attractive areas [30]. Another project called one million manga pages visualized 1,074,790 pages from 883 manga series, as seen in Figure 2.16. The images were organized according to visual characteristic with the aim of analyzing large cultural datasets [11].

2.2 Traditional Visualization Tools

There are many commercial systems available that can help in building a visualization. Spotfire by TIBCO software is used to extract important patterns or trends from large

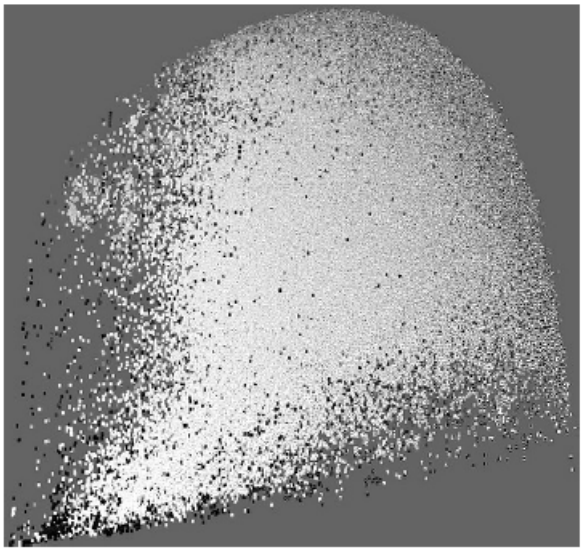

Figure 2.16: One million manga pages visualization [11]

datasets and is capable of rendering bar charts, line charts, 2D and 3D scatterplots, heat maps and network graphs [66]. It also supports distributed exploration so different users can explore and visualize in a common data set [2]. Spotfire helps users in preparing data and automating the visualization procedure [40]. However, it is very specific about the format of the data, and in order to write custom functions, one needs to learn Spotfire S+ [66].

Paraview also supports visualizing large datasets. It uses distributed computing to analyze datasets at a terascale level [39]. Paraview supports developing interactive 3D visualizations to explore volume data and can be programmed using Paraview's batch processing capabilities. It is supported by a very powerful community including domain researchers and visualization experts [54]. When the data to be visualized is very large, it becomes difficult and time consuming to move the data across a network. ParaViewWeb was developed to address this issue. Here, the visualization is built by a server and the images are streamed to the client using a web interface [27].

ArcGIS is a suit of software built for analyzing geo-spatial data [14]. It offers editing, querying, cluster analysis, visualization of geographic data and other similar functions. Many extensions are available for ArcGIS that can be downloaded to increase its functionality.

Another popular visualization tool is VisIt. It can be used to build interactive visualization with cross platform compatibility. It provides C++, Java and Python interfaces making it possible to add new visualizations to existing functionality [63]. VisIt divides the datasets into smaller pieces and renders them in parallel.

Gephi is a cross platform open source visualization tool [19]. It can be used to build real time visualizations for network analysis of graphs with up to 50,000 nodes and 1,000,000 edges. It provides a UI with the ability to load and save existing projects and allows interactive data-table views for quickly modifying the data.

More recently, web based visualization libraries have emerged as web technology has advanced. Most of these libraries use native web capabilities to draw visualizations and offer APIs to control how data is represented.

2.3 Web Visualization Tools

There are many pre-built visualization tools available that one can use to visualize data on a web page. Google Chart Tools supports developing many different types of dynamic charts. It can visualize data from simple line graphs to complex hierarchical tree maps with flexibility to customize their style and look. By utilizing web technologies like SVG, HTML5 and Javascript, it helps to create interactive visualizations for desktop and mobile platforms [61]. Flot is another Javascript library for building charts. It renders line and bar charts, and supports combining different charts that share data on the same axis

[15]. Raphael uses SVG and VML for drawing vectors graphics on a web page [50]. d3 is another popular Javascript library for visualizing data using SVG, HTML and CSS. It supports different types of visualizations including scatter plots, parallel co-ordinates, cartograms, bubble charts and other types of charts. It provides dynamic interactions and animations [5]. Highcharts is a specialized Javascript library for building interactive line, spline, column, pie, bar and other types of charts. It has a flexible API for drawing charts and can export to wide range of formats [3].

Apart from the web based visualization tools, there are frameworks available to build visualizations for a traditional desktop platform and deploy them on the web page. R is a popular statistical environment. It provides functionality to build complex graphics [48]. RApache can be used to develop web applications based on R [49]. Processing is another environment that is widely used to build visualization. It supports rendering 2D and 3D graphics with OpenGL integration. Projects built using processing can be deployed to run on a desktop or as a Java applet embedded into web page [45]. Processing.js was developed to allow code to run on any HTML5 compliant browser. Prefuse provides similar functionality. It offers various tools to build interactive visualizations and supports database connectivity, dynamic queries and animation. It was originally developed for Java but later released as Flare, an Actionscript library, to support deploying visualizations for Flash [44, 31].

Many Eyes by the IBM Research and IBM Cognos software groups is another powerful tool to build visualizations. It provides flexibility regarding the format used to store the data, and uses Java and Flash to generate visualizations [51]. Tableau public is another visualization tool [56]. It comes with many built-in charts, and provides drag and drop functionalities to visualize data. Tableau Public can import data from multiple files or servers. It can automatically generate a visualization deployed on the web page that

can be accessed directly by using its URL. It requires no coding for creating simple visualizations. If more advanced visualization technique is required with custom user interface, then Tableau Public may require extra effort.

2.4 Data Management

All visualizations are designed to represent data, so it is important to manage the raw data prior to visualization. One strategy is to use a database management system. There are four general types of database management systems:

1. Inverted List: Here, the typical mechanism of tables containing rows and columns are used to store data with the index acting like an inverted list pointing to the actual location on disk where relevant data can be found [23]. ADR, DATACOM/DB and ADABAS implement this approach and provide high performance in large database environments [23].

2. Hierarchical: Hierarchical systems use a parent-child model organized like a pyramid. Each child is linked to exactly one parent while a parent may be linked to multiple children [24]. Advantages of this type of systems include fast access and quick updates due to the tree like structure. However, adding new fields or records may require the entire database to be redefined [23, 24]. Famous hierarchical systems include IMS from IBM and Windows Registry from Microsoft.

3. Network: Networks system differ from hierarchical systems by allowing child elements to have multiple parents [23]. As linkages are possible between different entities they are more flexible but are limited by the number of connections allowed [24]. Networks systems have a complex query language and so can be intimidating

to new users [23]. Some famous database systems using this model include TurboImage, Integrated Database Management System (IDMS) and RDM Embedded.

4. Relational: Relational systems uses relations instead of hierarchies between entities. Data is stored as rows or records with key fields that uniquely identify a row and connect tables to one another [24]. There are many advantages of relational database systems including rigorous design, flexibility, ease of use and powerful data retrieval [23, 24]. Disadvantages include higher computational resources and slower responses on relational queries [23]. RDBMS dominates the market for GIS and data processing because of the availability of proprietary systems like Oracle, DB2 and Ingres [23].

As the amount of data increased, traditional relational database systems started facing problems and there arose a need to process data effectively with high IO performance. To account for this, NoSQL databases were created [21]. There are many advantages of NoSQL databases including fast read/write, low cost, mass storage and easy expandability. However, NoSQL does not support important SQL features like transactions and joins [21]. Mainstream NoSQL databases can be categorized as 1) Key-Value, 2) Column-oriented and 3) Document based [21]. Redis is a key-value database that loads the entire data into the physical memory for performing operations. Hence, it provides very good performance but its application is limited by the amount of physical memory present in the system [21]. Cassandra is a column oriented distributed database used by the Facebook. It is flexible in terms of database schema, supports range queries and is highly scalable [21]. MongoDB is a document database that supports complex data types, powerful queries and indexing. One of its biggest advantages is fast access to mass data, almost an order of magnitude faster than MySQL [21].

Chapter 3

Design

There are three important components of any visualization tool: 1) Visualization, 2) User Interface (UI) and 3) Data management. We shall discuss design rationale for these in detail in this chapter.

3.1 Visualization

There are many ways to display visualizations on the web page. One can have a server compute the visualization result, capture the produced image and send it over the network. By doing this, one can avert the need for high computational power on client devices. Also, if a visualization is transferred as an image, it can be easily displayed across a large number of devices. Other advantages include high performance if distributed computing is utilized, low memory consumption on the client side and access based on authorization. However, if multiple high quality images are required then it might be inefficient to transfer them. Moreover, controlling the visualization is difficult since each request has to be sent to the server which builds a new image, which is then

transferred back across the network.

It is now possible to draw custom geometry in a web page using Canvas, SVG and WebGL. With advances in HTML5, all modern browsers support these entities to some degree. Canvas was added as a part of HTML5. It is raster based and supports drawing graphics through Javascript. Modern browsers are now utilizing hardware acceleration to render Canvas objects. However, as Canvas renders graphics as pixels, it is hard to associate events with specific objects. Also, the entire canvas has to be redrawn while updating the image [20].

Scalable Vector Graphics (SVG) is also well known for drawing geometric objects on a web page. It follows an XML type syntax for creating hierarchies of elements which can be combined to form complicated shapes. As each SVG object is represented under a separate SVG DOM element, it is much easier to attach events to the objects. It is also easier to access the individual elements for content manipulation. SVG supports drawing points, lines, circles, ellipses, rectangles and polygons. It also provides color scales, transparency, patterns, the ability to render text, and a powerful event model. However, SVG can sometime led to slow performance when rendering large numbers of elements.

WebGL is a new technology for rendering 2D or 3D graphics using the Javascript. It is based on OpenGL ES 2.0, supports hardware acceleration and can be included in a Canvas element. WebGL allows direct access to the GPU making it fast and efficient for rendering graphics [10]. However, WebGL is a comparatively new technology and is not fully supported in all web browsers.

We used the d3 library for drawing visualizations on a web page. Since the library uses SVG for drawing objects, we decided to use SVG exclusively in our visualization. SVG is compatible with all modern browsers, so it does not depend on any external plugins

for drawing the objects.

3.2 User Interface (UI)

Once a visualization is displayed, some method of control is needed. Standard HTML UI elements include textboxes, buttons, checkboxes, menus, or radio buttons. Each of these UI elements can help users to steer the visualization according to their needs.

We wanted to have interactive elements to choose which data to visualize. One approach is to use the basic UI widgets built into HTML and CSS. Alternatively, there are many libraries like Dojo, Prototype, jQuery and Yahoo User Interface (YUI) that can help in building a more powerful UI.

We built our system's UI with existing libraries, avoiding the need to design advanced interactive elements from scratch. These libraries create a consistent and interactive user interface. The libraries are small in size making them an ideal choice for a client based visualization system.

Dojo is popular library that abstracts different types of browsers and provides a common API along with a set of user interface widgets like menus, tabs, tables, charts, gauges and calendars [16]. Even though Dojo provides a very powerful widget base, it has a steep learning curve compared to other libraries. It provides a very high level of abstraction that may not be necessary for developing a standard user interface.

Prototype is small in size, provides powerful selectors, supports AJAX calls and has an active community [46]. It allows widgets and effects but does not provide as extensive a set of widgets as other libraries.

jQuery is a Javascript based library [28]. It is cross-browser and cross-platform compatible and provides excellent flexibility for manipulating DOM elements. It is light-

weight, fast, helps in creating dynamic web pages, supports a wide range of events, is CSS3 compatible, and supports animations and AJAX calls. It has become an industry standard that various companies are using in their products. There are many libraries that use jQuery as a basic foundation, offering a wide range of functionality. jQuery also provides widgets for user interaction. It has a separate version for tablets and mobile devices that provides a touch-optimized framework.

YUI is a free and opensource library based on Javascript and CSS [70]. YUI offers many features to build an interactive UI for web applications. Controls include auto complete, calendar, charts, color picker, slider, tab view, dial, and uploader with progress bar.

For the NCBP project, a customized menu was built using jQuery, CSS and Javascript to allow a user to filter both data and attributes for visualization. The jQuery UI's tabs widget was used to provide a mechanism to switch between different functional components. We used a datepicker widget for selecting appropriate date ranges. Additionaly, a DataTable library built on the top of jQuery was used to display data in a sortable tabular format [57]. Using these existing libraries saved time and effort, allowing us to concentrate on development of the visualization. For the MMNC project, we chose to evaluate a different library: YUI. It provided two sliders, one for the year and one for unemployment rate. The combined value of the sliders filters the visualized data.

3.3 Data Management

As visualizations are built using data, a separate component dealing with managing, processing and retrieving data is helpful. There are two common choices when dealing with data: 1) Use a DBMS and 2) Use a flat file. DBMSs offer many data handling

functions. We looked into two types of databases: 1) SQL and 2) NoSQL. As we were dealing with geo-spatial data, a database system supporting geo-spatial queries is also desirable. MongoDB, Oracle and PostgreSQL were chosen as potential candidates. Oracle and PostgreSQL are SQL database while MongoDB is a NoSQL database. If DBMSs are used then response time can be potentially higher due to overhead produced by processing queries, indexing or unpacking data [23]. So, not using any DBMS may help in improving the response time but at the expense of losing functionality [23].

Oracle is widely used for many commercial level applications owing to maturity, excellent community support and reliability [9]. It falls under the object-relational database system. It is a multi-user database supporting the ACID (Atomicity, Consistency, Isolation, Durability) property. One major advantage of this database is the ability to form relationship and execute complex queries.

PostgreSQL is open source, free, mature, runs on major operating systems, supports many data types and provides interfaces to many programming languages [43]. It supports geographic objects through PostGIS, enabling it to perform geo-spatial queries for geographic information systems [42]. PostGIS can store advanced data types like points, polygons and lines. This supports the development of web services that need to perform spatial or geometric queries.

NoSQL databases like MongoDB offer flexibility to store unstructured records like text or images. It is easy to use, fast, supports indexing and easily integrates with other programming languages [67, 4]. NoSQL databases are commonly used in high-scale web applications as they are faster than relational databases for certain types of data [4]. MongoDB does not support traditional JOIN operations but provides functions like conditional updates, composite keys, text search, aggregations, map-reduce, sharding and geo-spatial searching [4]. MongoDB is a relatively new technology so certain capabilities

like security are still being improved.

For the NCBP project, we decided to use a DBMS because of the structure of the data. A database system that can store information about spatial regions and export it through web services was desirable. Querying and storing the data became easier with PostgreSQL. We used the ESRI shape file format to store boundary information and PostgreSQL to export it in GeoJSON format, which is used by d3 to visualize regions [13].

For the MMNC project, we decided not to use any DBMS. As we were not performing complicated queries on the data, we built a flat file residing on the server to provide data. This strategy saves time in building web services for retrieving data. There are many formats for storing data like CSV (Comma Separated Value), JSON (JavaScript Object Notation) and XML (Extensible Markup Language) [18, 29, 65]. Our data was stored in a CSV format which is parsed by d3. We also used ESRI shape files to define boundary information for North Carolina state counties. This was converted to a GeoJSON format, which is used by d3 for drawing the counties.

Chapter 4

Implementation

We used web based technologies to visualize data for the National Collaborative for Bio-Preparedness (NCBP) project and the Missing Meals of North Carolina (MMNC) project. Both the projects contained data with geo-spatial information. In NCBP, we visualized medical alerts obtained from different sources. In MMNC, we visualized missing meals data for North Carolina state obtained from Food Bank of Central and Eastern North Carolina and augmented by Dr. Aric Labarr, Assistant Professor at Institute of Advanced Analytics. In this chapter, we discuss the implementation of the systems in terms of visualization, user interface and data management.

4.1 National Collaborative for Bio-Preparedness

NCBP is a combined partnership involving North Carolina State University, SAS Institute, UNC-Chapel Hill and Department of Homeland Security. The primary goal is to provide a state-wide system to improve health security by focusing on:

1. Enhanced surveillance.

2. Improved situational awareness using advanced analytics.

3. Improved confidence in informative systems.

4. Improved health care and management.

4.1.1 Architecture

Figure 4.1 shows the architecture of the NCBP system. It consists of a database server, and two web applications hosted on an application server. The database server manages two databases:

1. *Visualization*: Holds alerts data, and geographic information for different regions.

2. *User*: Holds user information.

The application server hosts two web applications:

1. *Visualization APIs*: Provides different types of web services to access alerts data and geographic information from the *Visualization* database.

2. *User application*: Acts as an authentication portal for the users to access the web services offered by *Visualization APIs*.

4.1.2 Data Management

Raw data was provided in CSV format with each record containing multiple attributes. Important attributes for each alerts include: 1) Date, 2) County 3) Latitude and Longitude 4) Type of Event 5) Source 6) Zip code and 7) Analytical method. The data was collected from various sources including the Poison Control Centre (PCC), the Emergency

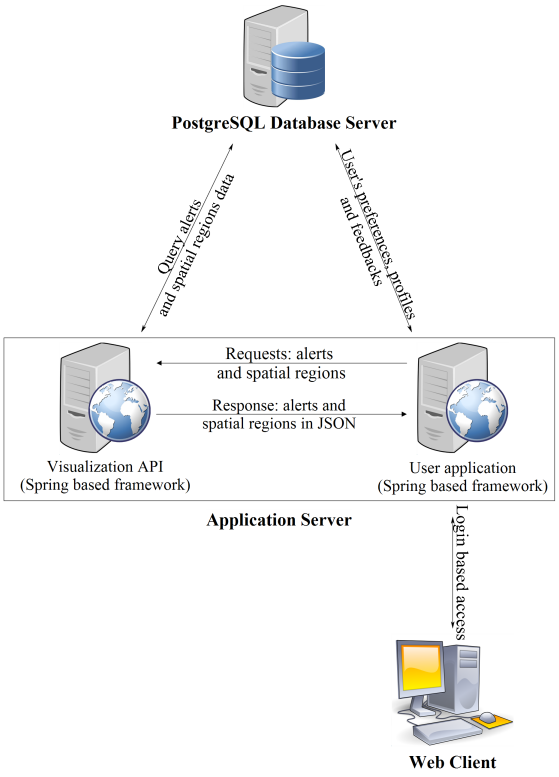

Figure 4.1: NCBP architecture

Medical Service (EMS) and the Emergency Department (ED). We also used population counts and shape files of different regions (zip-3, zip-5, county and congressional district) obtained from the US Census Bureau.

We stored data in a PostgreSQL database. The REST (REpresentational State Transfer) architecture is used to communicate with the database server and retrieve data in JSON format. REST has several advantages including improved performance, enforced security and interface simplicity. We stored the alerts data and the shape files of different

regions in the *Visualization* database. This boundary information is also assigned a region code. Thus, boundary information can be easily obtained by querying the database. We set up web services that take in a region code and return boundary data. We also setup web services to return alerts data over a user specified date range. The geo-spatial analysis features of PostGIS and PostgreSQL were used to perform location based queries. Region boundaries were simplified to produce smaller boundary descriptions. For mapping alerts to specific regions, we used a point in polygon feature of PostGIS.

4.1.3 Project Components

The NCBP project uses login based authentication to provide access to the visualization tool and web services. There are five major components in this project:

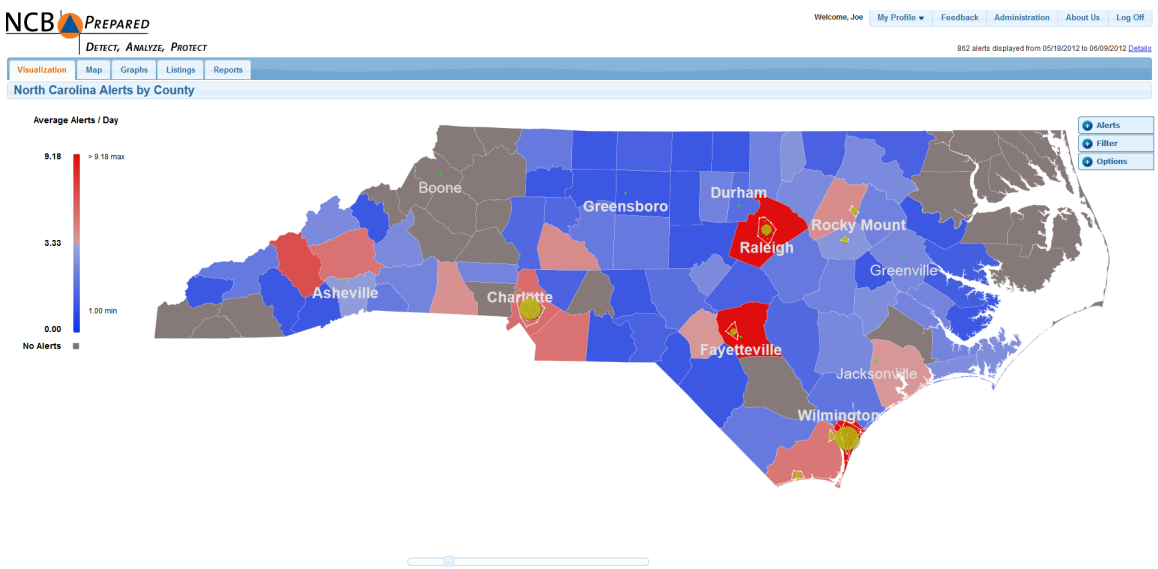

Figure 4.2: Visualization tab to represent medical alerts on North Carolina state

1. Visualization: This component aggregates alerts and visualizes them as a choropleth map. It also contains a user interface to control the visualization. Figure 4.2 shows the visualization tab with the alerts data represented as a red-blue choropleth map.

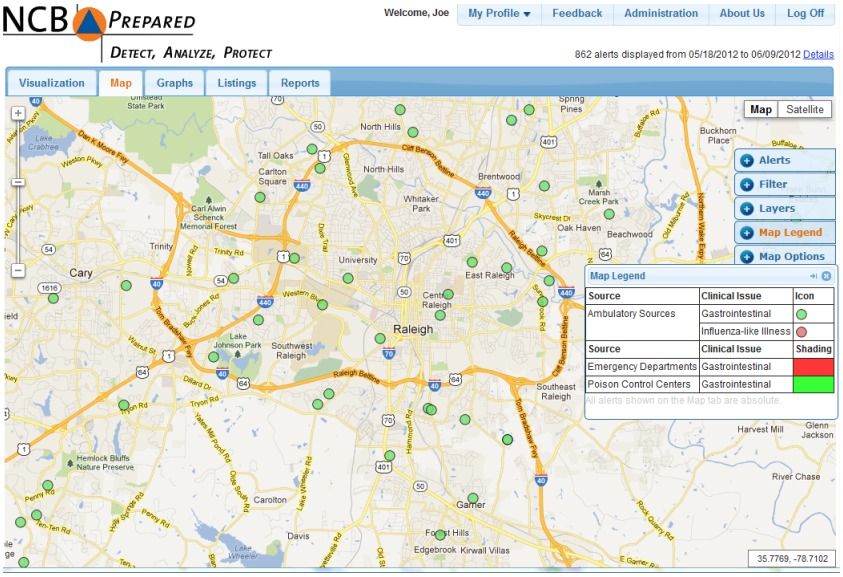

Figure 4.3: Map tab visualizing alerts on google maps in Raleigh area

2. Map: This component visualizes each individual alert using Google maps. It can help to correlate possible causes of alerts based on geographic location from the surroundings. Figure 4.3 shows the map component of the project with colored dots representing different types of alerts.

3. Graphs: This component visualizes alerts as bar graphs, pie charts and timelines (Figure 4.4).

4. Listings: This component lists alerts as text descriptions within a user chosen date range. As seen in Figure 4.5, this allows detailed inspection of each alert's attributes.

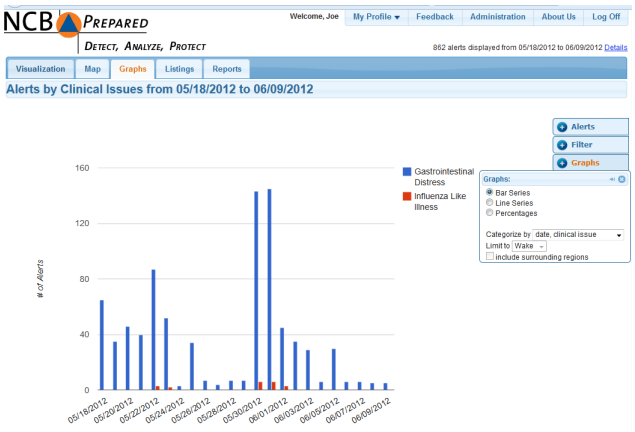

Figure 4.4: Graph tab with alerts visualization through a bar graph

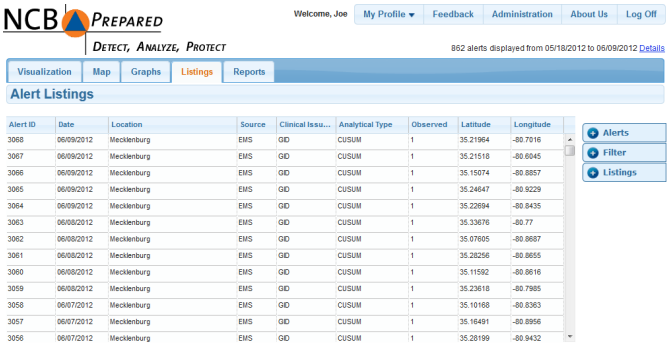

Figure 4.5: List tab with alert's attributes displayed in a tabular format

5. Reports: This component summarizes the results of the visualization. It provides an overview of the alerts that can be printed and emailed to other users for analysis.

4.1.4 Visualization

Our work concentrated on the visualization and report components. These components allows a user to compare alerts across different geographic regions. We used a choropleth mapping technique where the regions are shaded according to the number of alerts generated. As seen in Figure 4.6, we used a bi-color coding model with red on the upper range

and blue on the lower range. The colors were chosen to be perceptually balanced. This ensures a common value difference produces an equal change in perceived color difference everywhere across the color scale.

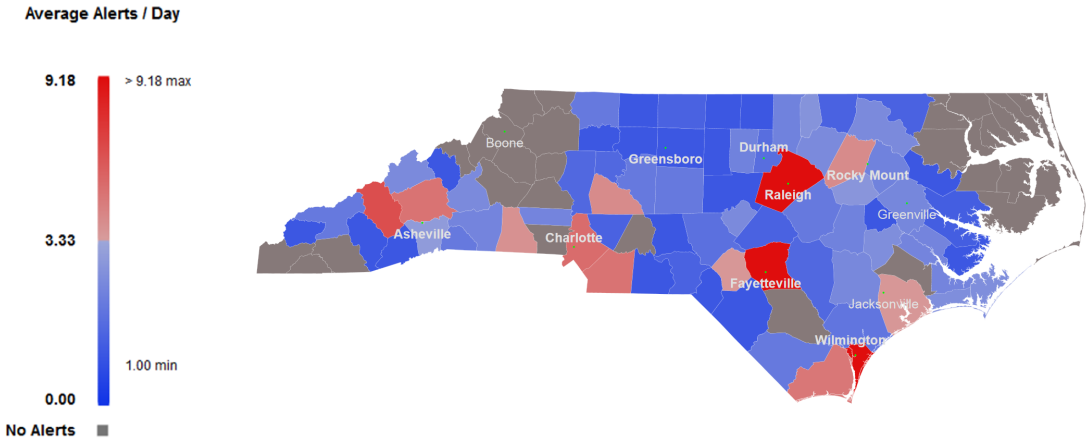

Figure 4.6: Choropleth map visualizing alerts for county regions

For mapping the alerts counts to an (r, g, b) value, we calculate the average number of alerts per day in all regions. We set the upper bound of the color scale to two standard deviation above the average and lower bound to two standard deviation below the average. This ensures that outliers do not compress the color scale, which would cause the majority of the alerts to be visualized with nearly identical colors. We define the upper and lower bounds of the color scale *max* and *min*, respectively, and the average alerts per day *avg*. Regions shaded red have an alert count higher than or equal to *avg*, and regions shaded blue have a count below *avg*.

When a date range is selected, we calculate the average alerts per day avg_v for each region within the selected range. Eq. 4.1, Eq. 4.2 and Eq. 4.3 shows the formula used to calculate (r, g, b) when avg_v is greater than or equal to *avg*.

$$r = 215 + \left(\left(\frac{8}{max - avg}\right) * (avg_v - avg)\right) \tag{4.1}$$

$$g = 156 - \left(\left(\frac{143}{max - avg}\right) * (avg_v - avg)\right) \tag{4.2}$$

$$b = 156 - \left(\left(\frac{143}{max - avg}\right) * (avg_v - avg)\right) \tag{4.3}$$

Similarly, Eq. 4.4, Eq. 4.5 and Eq. 4.6 shows the formula used to calculate (r, g, b) when avg_v is less than avg.

$$r = 158 - \left(140 * \left(1 - \frac{avg_v}{avg}\right)\right) \tag{4.4}$$

$$g = 167 - \left(114 * \left(1 - \frac{avg_v}{avg}\right)\right) \tag{4.5}$$

$$b = 217 + \left(14 * \left(1 - \frac{avg_v}{avg}\right)\right) \tag{4.6}$$

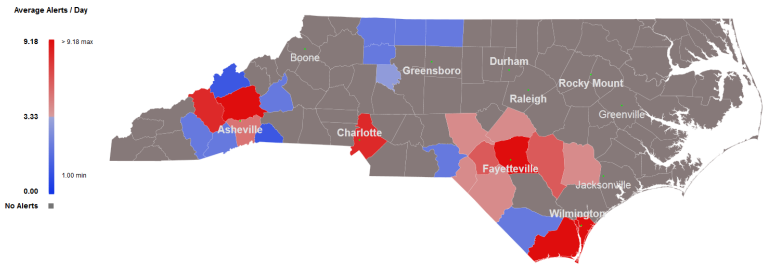

Figure 4.7: Counties having spikes in daily alert activity

We also capture the maximum and minimum average alerts per day for the selected range and display them on the color scale. We define these values *local max* and *local min* respectively. This allows a user to see how the maximum and minimum alert counts for the given time range compare to the overall *max* and *min* for the entire dataset. Figure 4.7 shows a visualization of average alerts per day from 01/07/2012 to 02/15/2012. The counties having the strongest reds produced unusual spikes in the reported alerts. Figure 4.8 shows the color scale for average alerts per day along with the *local max* and the *local min*. The *avg* value is represented between the red and blue gradients.

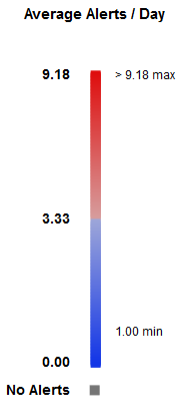

Figure 4.8: Colored scale for NCBP

Regional population counts are available to normalize the data to average alerts per day for every 10,000 people. Visualizing alerts counts without considering population count might produce a biased result with more alerts in highly populated regions. Figure 4.9 and Figure 4.10 shows a visualization using absolute and normalized alerts for the same date range. Wake County is one of the most populated counties in North Carolina. It is red when absolute alerts are visualized but turns blue when normalized alerts are

shown.

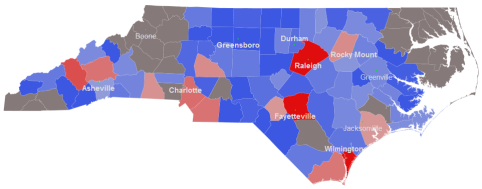

Figure 4.9: Absolute alerts

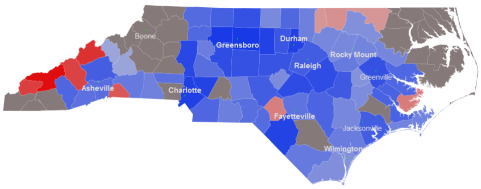

Figure 4.10: Normalized alerts

We used the position of each alert to build spatial clusters of alerts. A cluster organizes entities into groups based on similarity, in our case position similarity. We built clusters of alerts using a minimum spanning tree (MST) technique. A spanning tree of a graph $G = (V, E)$ is a sub graph that contains all the vertices and forms a tree. The MST is a spanning tree with a total edge weight less than or equal to all other spanning trees. An MST of an undirected graph can be calculated from Prim's, Kruskal's or Boruvka's algorithm. We used Kruskal's algorithm to compute an MST as follows:

1. Construct a forest F of trees with each vertex of the graph G as a distinct tree.
2. Construct a set S that contains all the edges of the graph G.
3. While S is not empty and F is not spanning
   (a) Delete an edge having a minimum weight.
   (b) If that edge connects two trees forming a single tree, then add it to F otherwise discard it

After obtaining an MST, a threshold value is selected that throws away unwanted

edges resulting in one or more connected sub-components. These components are returned as clusters. We calculate clusters from alert points v_0 to v_n as follows:

1. Construct an undirected graph G with edges from v_0 to all other nodes.
2. Sort the remaining vertices from v_1 to v_n by x-position and store them in set S (Optimization step).
3. Construct an edge list between all the vertices in set S and add those edges having edge weights within some predetermined or user defined threshold to G.
4. Construct an MST using Kruskal's algorithm from G.
5. Remove all edges from the MST having edge weights more than some predetermined or user defined threshold.
6. Find the resulting connected sub-components and return them as individual clusters.

Here, we used Euclidean distance between two alerts to weight their edges. Our algorithm returns a list of clusters and their associated alerts. We find the centroid of a cluster by using the center of mass of its alerts. Using SVG, it is possible to draw clusters as circles. We used d3's Mercator projection to convert the cluster's latitude-longitude center to an appropriate pixel location (x, y). Figure 4.11 shows clusters placed on top of the choropleth map using yellow circles. The clusters help to identify local areas with a large number of alerts. In multiple cases, we found clusters in regions having spikes in the daily alert activity data.

We visualized the number of alerts in a cluster by varying its circle size. More alerts generate large circles. We chose an appropriate opacity level for the cluster circles to

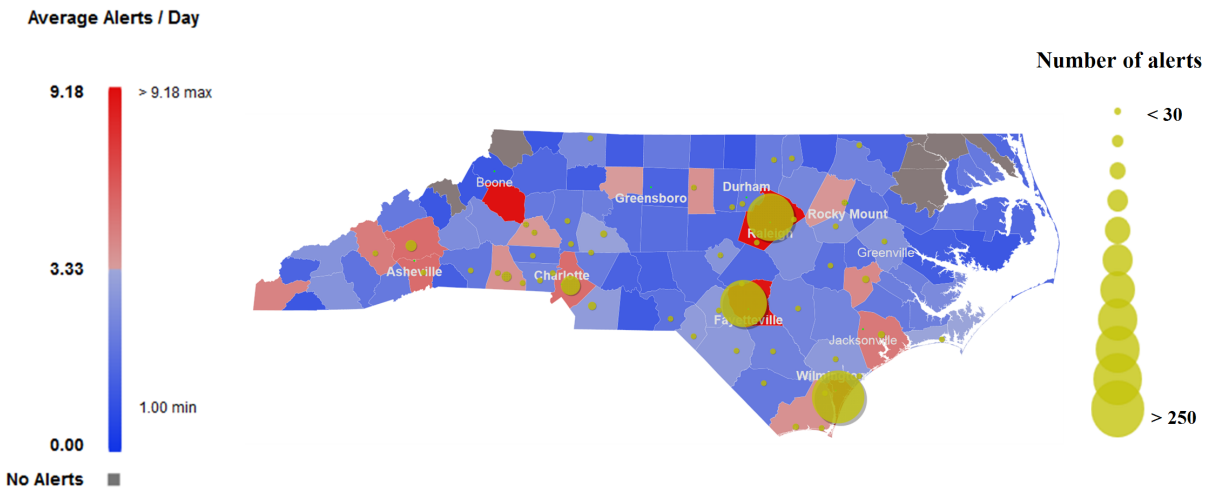

Figure 4.11: Choropleth map with clusters

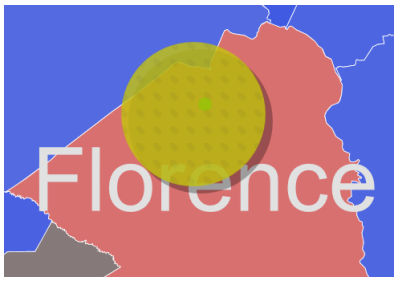

Figure 4.12: Clusters with "float" effect

allow viewers to "see" the region beneath it. We also made the clusters appear to "float" over the map by adding an offset drop value (Figure 4.12).

To show where an alert falls in the user chosen date range, we calculate the difference between an alert's date and the date of the first alert in the cluster. Large differences (i.e., older alerts) are visualized with an increased opacity. Thus, as seen in Figure 4.13, highly visible alerts are ones that occurred farthest from the cluster's earliest alert.

Displaying clusters of alerts as a circle gives no information about the alerts exact

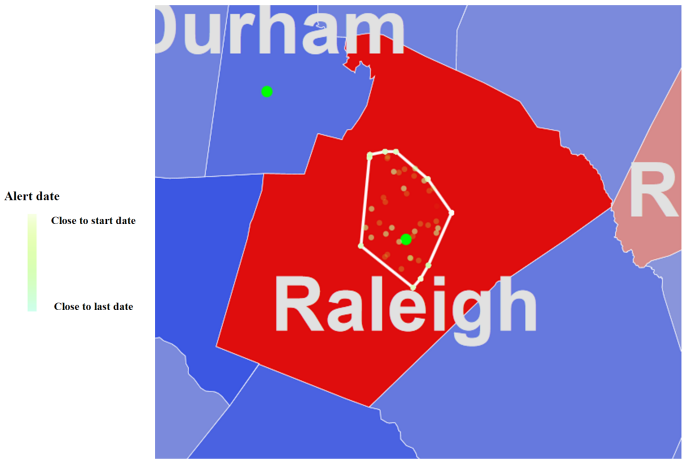

Figure 4.13: Visualizing date attribute of alerts

spatial locations. To address this, we built a convex hull using alerts position as input to approximate the organization of the alerts cluster. The convex hull for a set of points P is the boundary of the smallest convex region containing P [33]. We used an existing library for building the convex hull [58]. Figure 4.14 shows the convex hull formed from a cluster of alerts. The convex hull can provide clues about how alerts within a cluster appeared over time (Figure 4.15 and Figure 4.16).

Figure 4.14: Convex hull within clusters with alerts displayed as points

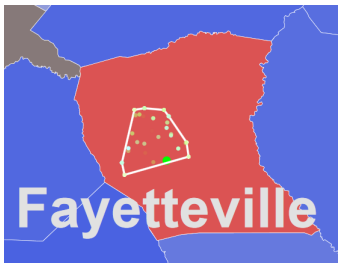

Figure 4.15: Alerts displayed from 01/01/2011 to 03/01/2011

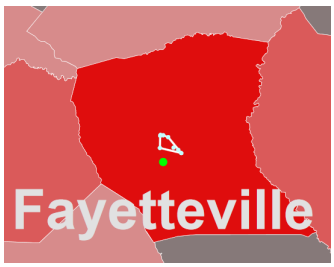

Figure 4.16: Alerts displayed from 01/01/2012 to 03/01/2012

Variance can be used to determine dispersion of data samples from the mean. We estimated the distribution of the number of alerts per day that lie in a cluster using variance. A low variance indicates that the cluster's alerts arrive at a fairly consistent rate across the date range. High variance suggests spikes in daily alert activity. We represent variance by layering a texture pattern on the top of a cluster. High variance is visualized with increased opacity of the texture (Figure 4.17).

Figure 4.17: Variance visualization using texture patterns

Initially, the NCBP project supported alerts for the state of North Carolina. As we had generalized components in place for drawing the visualization, we later extended it

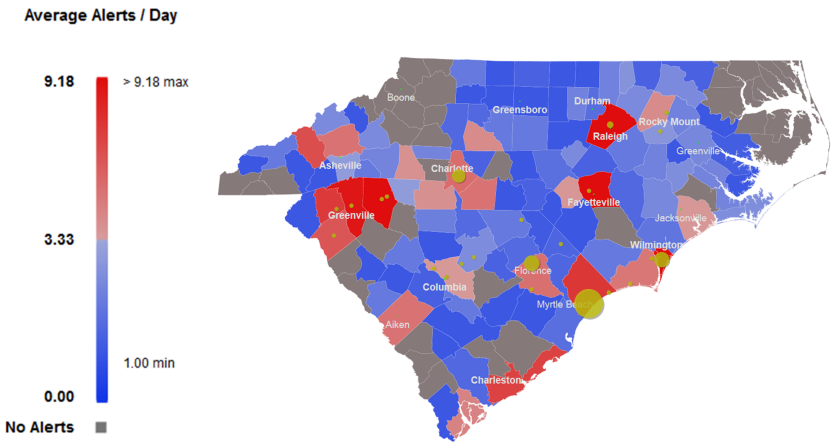

Figure 4.18: North Carolina and South Carolina alerts visualization with counties

to support South Carolina. We acquired the shape files, the population counts and the alerts data for South Carolina, then visualized both the states simultaneously by querying the database (Figure 4.18) We also obtained shape files of zip-3, zip-5 and congressional district regions for North Carolina and South Carolina, and extended our tool to support visualizing alerts on these types of regions (Figure 4.20-Figure 4.25). Figure 4.19 shows the alerts mapped on congressional districts for 2 of the 48 contiguous states of USA.

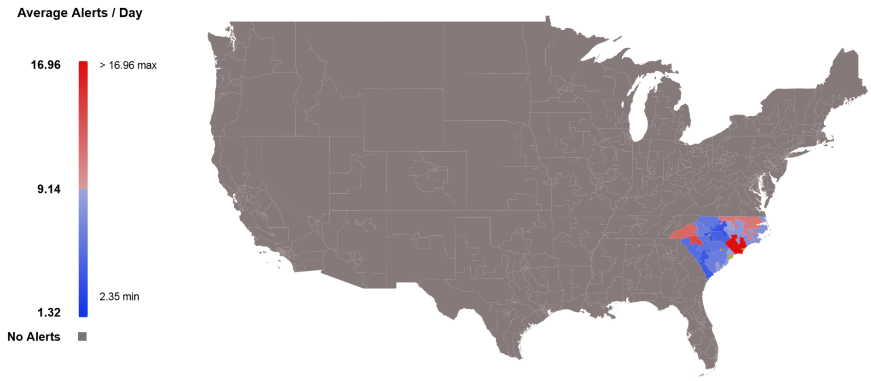

Figure 4.19: USA alerts visualization with congressional districts

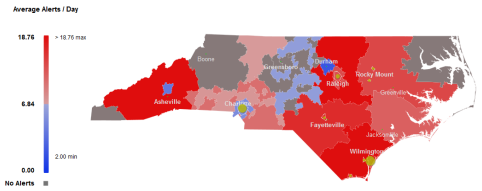

Figure 4.20: North Carolina alerts by zip-3

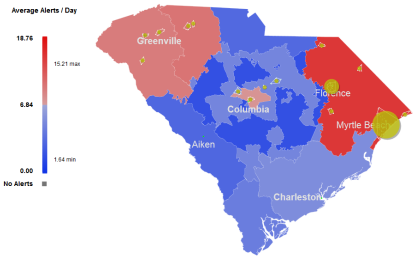

Figure 4.21: South Carolina alerts by zip-3

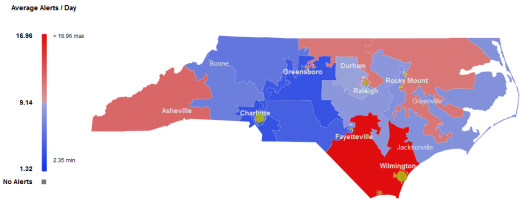

Figure 4.22: North Carolina alerts by congressional district

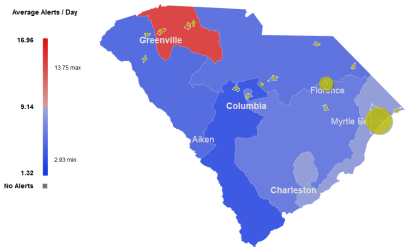

Figure 4.23: South Carolina alerts by congressional district

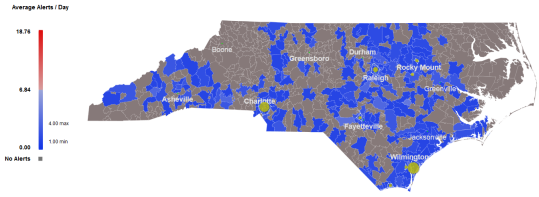

Figure 4.24: North Carolina alerts by zip-5

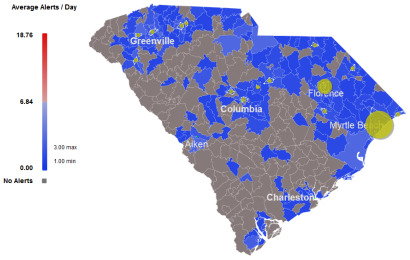

Figure 4.25: South Carolina alerts by zip-5

4.1.5 User Interface

We used jQuery to build an interactive user interface. jQuery UI's Tab widget was used to organize functional components in the system. We used a collapsed filter menu to offer better visibility on the visualization with check boxes for filtering different attributes of the visualized data and drop down menus for choosing boundaries of different regions.

Check boxes are also used to select or unselect convex hulls and clusters. Since we calculate clusters on the client side, we default to turning them off to optimize the load time of the visualization. We also provide the option to enable or disable normalization through population using radio buttons. We used a datepicker and a free form text field for selecting the date range of the visualized data.

While developing the UI, we used AJAX calls to make the interaction efficient. AJAX prevents the need to reload a page in order to display new content. By modifying the DOM elements using jQuery, it is possible to update content dynamically. We attached AJAX calls to all filtering options in our menu. This helps to dynamically update the visualization whenever a user interacts with the menu to filter the data.

Within the visualization itself, we display detail information using tool tips when a mouse is hovered over regions, textures or clusters. The entire visualized image can be zoomed using jQuery's slider or the mouse scroll wheel or dragged to reposition the map. This is useful when the screen size is too small to accommodate the entire image.

4.2 Missing Meals of North Carolina

This project was developed in association with the Food Bank of Central and Eastern North Carolina, a nonprofit organization that has provided food to people in 34 North Carolina counties for 30 years [37]. In this project, we visualized missing meals: the

number of meals low-income people are likely to miss because they cannot afford them. The goal for developing this visualization is to create awareness within the community and help in raising funds to continue provision of such meals.

4.2.1 Architecture

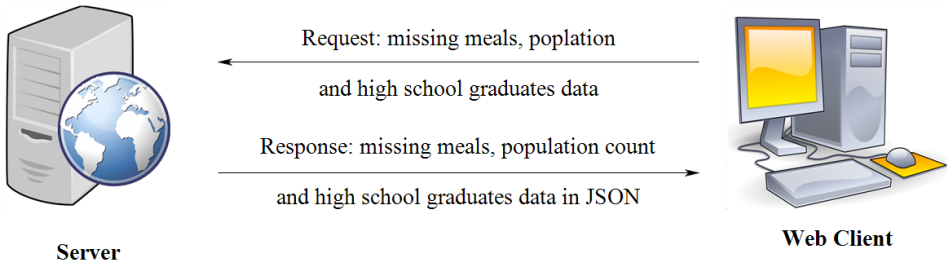

Figure 4.26: MMNC architecture

Figure 4.26 shows the architecture of the MMNC project. It consists of a client and a server. The client connects to the server, obtains the data and draws it as a visualization. A DBMS is not used due to simplicity of the data. This saved some effort in building and managing communication, but at the expense of less flexibility and potentially less capability if our data becomes more complex in the future.

4.2.2 Data Management

Missing meals data from the Food Bank was augmented by Dr. Aric Labarr, Assistant Professor at Institute of Advanced Analytics, to estimate the number of missing meals for different counties for the years 2007 to 2010 at unemployment rates of 5% to 25%

in 2.5% steps. The data was provided in CSV format. We obtained population data and ESRI shapefiles for each county from the US Census Bureau. The shapefile boundaries were converted into a GeoJSON format that can be parsed by d3. We also obtained the percentage of high school graduates in each county for the years 2007 through 2010, allowing us to correlate this data with estimated missing meals. We uploaded all of our data to a web server, then used d3 to read and parse it.

4.2.3 Visualization

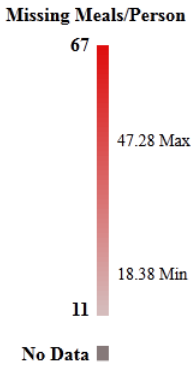

Figure 4.27: Colored Scale for MMNC

In the MMNC project, we visualized missing meals in each county with a uni-color scale. We decided to select different shades of red to represent the number of missing meals. Comparing counties based on the absolute number of missing meals may not be useful as counties with high populations may lead to higher number of missing meals. So, we normalized the missing meals by population to address this issue. We then identify the highest and lowest missing meals per person over the entire dataset and mapped

them to the highest and lowest saturated shades of red. We call these two values *max* and *min*, respectively. Similar to the NCBP project, we also annotate the color scale with the minimum and maximum missing meals per person for the data currently selected. We call these values *local min* and *local max*. Figure 4.27 shows our color scale. Higher missing meals per person results in a stronger red color. Eq. 4.7, Eq. 4.8 and Eq. 4.9 show the formulas used to calculate the (r, g, b) color value assigned to a counties for x missing meals per person.

$$r = 213 + \left(\frac{10}{max - min}\right) * (x - min) \tag{4.7}$$

$$g = 192 - \left(\frac{178}{max - min}\right) * (x - min) \tag{4.8}$$

$$b = 192 - \left(\frac{178}{max - min}\right) * (x - min) \tag{4.9}$$

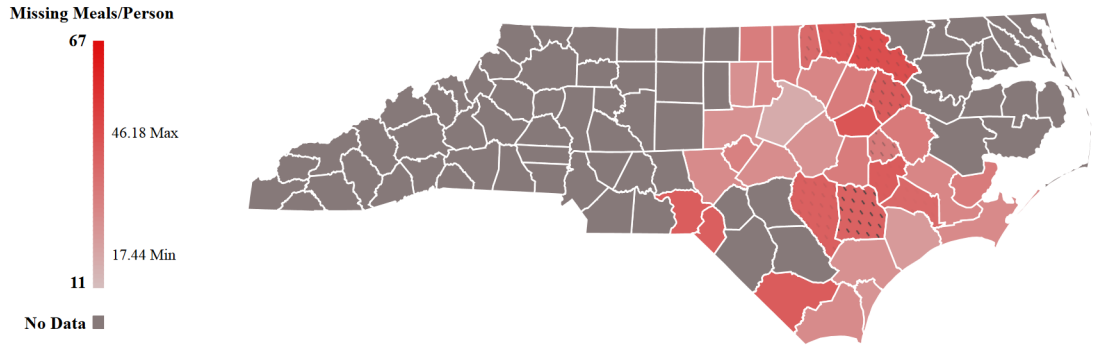

Figure 4.28: Visualization of missing meals per person at 5% unemployment for the year 2007

Figure 4.28 shows a visualization based on missing meals data. Sometimes, changes in the number of missing meals are small when the filtering parameters unemployment rate and year are changed, making it difficult to see differences in the map. The *local min* and *local max* values also update, making it easier to identify small changes at the county level. Keeping the year parameter constant and changing the unemployment rate identified less influence than expected on the missing meals (Figure 4.29-Figure 4.32). Keeping the unemployment rate constant and changing the year identified a sharp increase in missing meals during the recession year of 2008. Missing meals then gradually decreased as federal contribution to SNAP (Supplemental Nutrition Assistance Program) were dramatically increased (Figure 4.33-Figure 4.36).

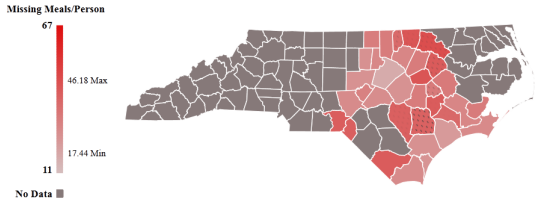

Figure 4.29: Missing meals at 5% unemployment for the year 2007

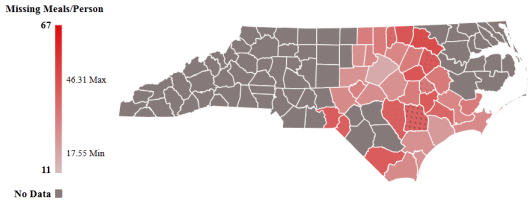

Figure 4.30: Missing meals at 7.5% unemployment for the year 2007

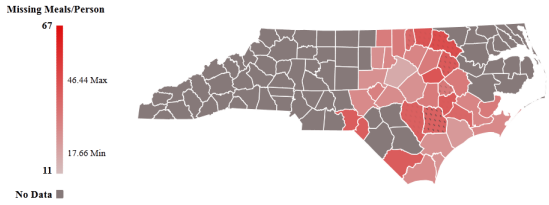

Figure 4.31: Missing meals at 10% unemployment for the year 2007

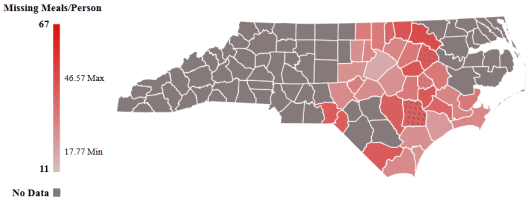

Figure 4.32: Missing meals at 12.5% unemployment for the year 2007

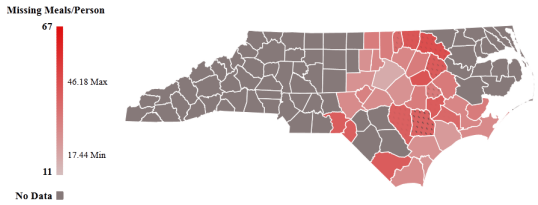

Figure 4.33: Missing meals at 5% unemployment for the year 2007

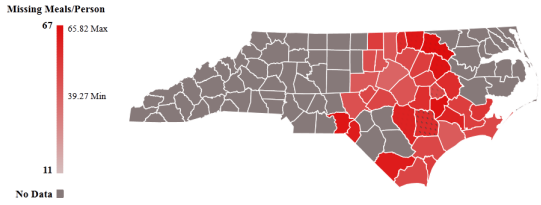

Figure 4.34: Missing meals at 5% unemployment for the year 2008

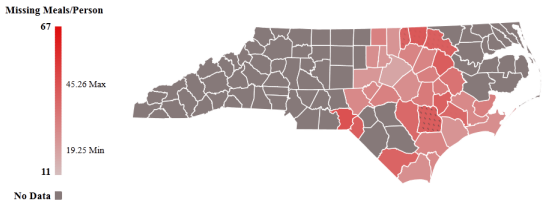

Figure 4.35: Missing meals at 5% unemployment for the year 2009

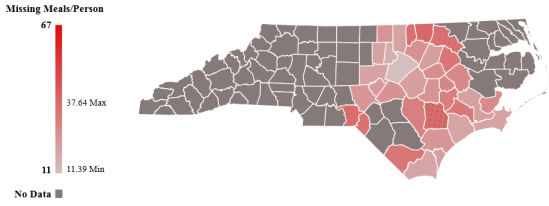

Figure 4.36: Missing meals at 5% unemployment for the year 2010

To try to search for predictions of missing meals, we visualized the percentage of people in a county that did not finish high school by adding a texture pattern. We used d3 to access the county's regions, then applied SVG's pattern feature to fill that area. Counties with highly opaque textures represents a higher percentage of people who did not finish high school (Figure 4.37). This highlighted multiple counties with higher missing meals and lower high school graduation rates. Any attribute of interest could control texture and opacity (e.g: poverty rate, household ownership rate and so on) allowing us to correlate against it.

4.2.4 User Interface

We explored a different library for developing a UI for the MMNC project. We used two tick sliders from the YUI library to interact with the visualization. The slider's position filters the data dynamically, allowing missing meals to be compared across subsequent

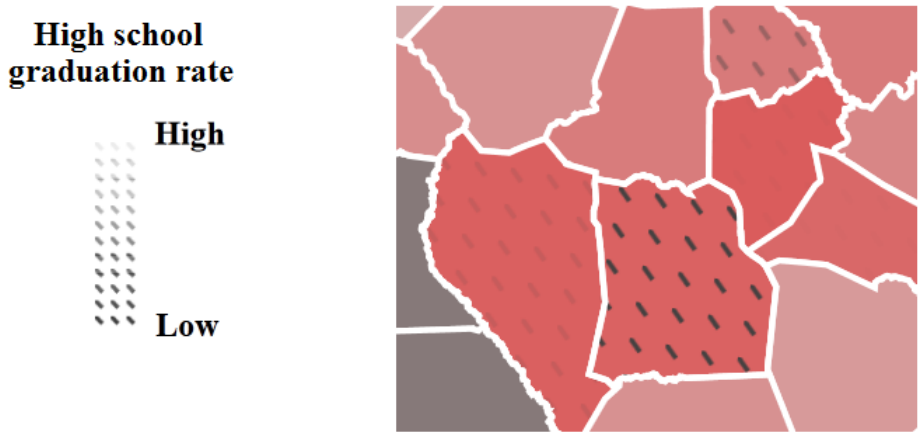

Figure 4.37: Visualizing high school graduation rates using texture patterns

years, and at different unemployment rates. Figure 4.38 shows the tick slider widgets used for selecting data.

We also offered a zoom feature attached to the mouse scroll wheel to analyze different parts of the visualization in more detail. The visualization can be dragged to change its position on screen. Double clicking on the visualization zooms in, centered about the click position.

Figure 4.38: Tick sliders to control unemployment rate and year

4.3 Evaluation

We used expert feedback to evaluate our visualization systems. Our users offered numerous suggestions and recommendations to design and improve the visualization for their

specific interests and analysis tasks. Each project went through multiple iterations to re-design both the visualization and the UI.

For the NCBP project, the experts agreed the visualization allowed for effective comparison of alerts between different regions. The clusters identified areas generating unusually high numbers of alerts, and the convex hull showed the alert's boundaries and progress over time. The experts also liked the ability to use the visualization tool on non-desktop platforms.

For the MMNC project, we visualized missing meals for different combinations of year and unemployment rate, with a prediction variable of high school graduation rate overlaid on each county. The experts quickly pointed out the sharp increase in missing meals during the recession year of 2008. They also concluded that missing meals are less sensitive to unemployment rate than they anticipated. They found that in a majority of cases counties having more missing meals were the ones having a lower high school graduation rate. Other contributing factors are suspected and we plan to visualize these in future versions of the tool.

We measured the rate at which the visualized image is refreshed in terms of frames per second (FPS) when translated on screen. Opera produced highest FPS. Chrome and Internet Explorer 9 performed better than Firefox and Safari. We suspect this result is due to the use of graphics hardware acceleration. We found that current web technology is flexible enough to accommodate a dynamic 2D visualization tool. It offered a powerful platform for building and deploying visualizations, resolving many issues regarding portability, accessibility and distribution.

Chapter 5

Limitations and Future Work

There were a number of limitations we encountered while developing the visualization tools. d3 provides a fixed API to its functionality. Fortunately, since d3 uses SVG for rendering, it is possible to manipulate its SVG objects directly. We used this approach to implement features such as textures, clusters and alert points.

It is difficult to get identical browsing experience, since the time needed to display a visualization depends on the size of the data, network's speed and computational capabilities of the device. Also, some browsers use graphics hardware acceleration to render SVG graphics, while others do not.

On a traditional desktop browser, certain UI operations (e.g: hovering) are available that do not exist on touch based devices. Also, devices having smaller size and resolution produces smaller versions of visualization that can be more difficult to use.

In this thesis, we developed two visualization tools using 2D web technologies. We do not offer any functionality to change the colors of the visualized image or import data. We do not perform any automatic analysis of the visualized data. The user is expected to manually inspect the visualization and use visual perception for analyzing the data.

We plan to extend our work by studying how to provide a complete visualization experience on non-desktop platforms. For the NCBP project, we plan to visualize all the medical alerts in the USA to offer a surveillance at national level. We also plan to include crime data to correlate with the medical alerts. For the MMNC project, we plan to correlate missing meals with poverty, age, gender and race.

Chapter 6

Conclusion

This thesis describes the use of web technologies to develop visualization tools. There are many issues in developing a visualization tools using web technologies such as selecting a suitable visualization drawing technology, building an interactive UI and managing data. We used web based technologies like SVG, HTML, Javascript and existing libraries to develop visualization for the National Collaborative for Bio-Preparedness (NCBP) project and the Missing Meals of North Carolina (MMNC) project. These technologies are supported on all modern browsers and the developed visualization can be accessed from desktop, tablets or mobiles devices. We explored two user interface libraries to build controls for interacting with the visualizations. For managing raw data, we used a database management system for the NCBP project and a flat-file based approach for the MMNC project. We obtained positive feedbacks for the developed systems suggesting that web technologies can be used to build visualization tools for the analytical support.

REFERENCES

[1] Rod Adkins. http://www-03.ibm.com/press/us/en/pressrelease/37920.wss.

[2] Christopher Ahlberg. Spotfire: an information exploration environment. *SIGMOD Rec.*, 25(4):25–29, December 1996.

[3] Highsoft Solutions AS. http://www.highcharts.com/products/highcharts.

[4] A. Boicea, F. Radulescu, and L.I. Agapin. Mongodb vs oracle – database comparison. In *Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on*, pages 330 –335, sept. 2012.

[5] Michael Bostock. http://d3js.org/.

[6] Michael Bostock, Shan Carter, and Matthew Ericson. http://www.nytimes.com/interactive/2012/09/06/us/politics/convention-word-counts.html.

[7] Michael Chau. Visualizing web search results using glyphs: Design and evaluation of a flower metaphor. *ACM Trans. Manage. Inf. Syst.*, 2(1):2:1–2:27, March 2011.

[8] Jason Chuang, Christopher D. Manning, and Jeffrey Heer. Without the clutter of unimportant words: Descriptive keyphrases for text visualization. *ACM Trans. Comput.-Hum. Interact.*, 19(3):19:1–19:29, October 2012.

[9] Oracle Database. http://www.oracle.com/us/products/database/overview/index.html.

[10] Daniel Davis. http://www.nzarchitecture.com/blog/index.php/2011/05/16/html5-webgl/.

[11] Jeremy Douglass, William Huber, and Lev Manovich. http://softwarestudies.com/Understanding_scanlation_2011.pdf.

[12] Matt Edmunds, Robert S. Laramee, Guoning Chen, Nelson Max, Eugene Zhang, and Colin Ware. Surface-based flow visualization. *Computers & Graphics*, 36(8):974 – 990, 2012. Graphics Interaction: Virtual Environments and Applications 2012.

[13] ESRI. http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

[14] ESRI. http://www.esri.com/software/arcgis.

[15] FLOT. http://www.flotcharts.org/.

[16] The Dojo Foundation. http://dojotoolkit.org/.

[17] George Gagaudakis. Content-based image visualization. In *Proceedings of the International Conference on Information Visualisation*, IV '00, pages 13–, Washington, DC, USA, 2000. IEEE Computer Society.

[18] GDAL. http://www.gdal.org/ogr/drv_csv.html.

[19] Gephi. https://gephi.org/.

[20] Robert Gravelle. http://www.htmlgoodies.com/html5/other/html5-canvas-vs.-svg-choose-the-best-tool-for-the-job.html#fbid=pzx4lFwv9ra.

[21] Jing Han, E. Haihong, Guan Le, and Jian Du. Survey on nosql database. In *Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on*, pages 363 –366, oct. 2011.

[22] Christopher Healey. http://www.csc.ncsu.edu/faculty/healey/tweet_viz/tweet_app/.

[23] R. G. Healey. http://www.wiley.com/legacy/wileychi/gis/Volume1/BB1v1_ch18.pdf.

[24] Gary Heberling. http://www.personal.psu.edu/glh10/ist110/topic/topic07/topic07_06.html.

[25] Ingrid Hotz, Louis Feng, Hans Hagen, Bernd Hamann, Kenneth Joy, and Boris Jeremic. Physically based methods for tensor field visualization. In *Proceedings of the conference on Visualization '04*, VIS '04, pages 123–130, Washington, DC, USA, 2004. IEEE Computer Society.

[26] Alfred Inselberg. http://astrostatistics.psu.edu/su06/inselberg061006.pdf.

[27] Julien Jomier, Sebastien Jourdain, Utkarsh Ayachit, and Charles Marion. Remote visualization of large datasets with midas and paraviewweb. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 147–150, New York, NY, USA, 2011. ACM.

[28] jQuery Project. http://jquery.com/.

[29] JSON. http://www.json.org/.

[30] Slava Kisilevich, Milos Krstajic, Daniel Keim, Natalia Andrienko, and Gennady Andrienko. Event-based analysis of people's activities and behavior using flickr and panoramio geotagged photo collections. In *Proceedings of the 2010 14th International Conference Information Visualisation*, IV '10, pages 289–296, Washington, DC, USA, 2010. IEEE Computer Society.

[31] UC Berkeley Visualization Lab. http://flare.prefuse.org/.

[32] David H. Laidlaw, Eric T. Ahrens, David Kremers, Matthew J. Avalos, Carol Readhead, and Russell E. Jacobs. Visualizing diffusion tensor images of the mouse spinal cord. Technical report, Pasadena, CA, USA, 1998.

[33] Runzong Liu, Bin Fang, Yuan Yan Tang, Jing Wen, and Jiye Qian. A fast convex hull algorithm with maximum inscribed circle affine transformation. *Neurocomputing*, 77(1):212 – 221, 2012.

[34] James C. Moore. Visualizing with vtk. *Linux J.*, 1998(53es), September 1998.

[35] Sebastian Moser, Patrick Degener, Roland Wahl, and Reinhard Klein. Context aware terrain visualization for wayfinding and navigation. *Computer Graphics Forum*, 27(7):1853–1860, 2008.

[36] G. P. Nguyen and M. Worring. Interactive access to large image collections using similarity-based visualization. *J. Vis. Lang. Comput.*, 19(2):203–224, April 2008.

[37] Food Bank of Central and Eastern North Carolina. http://www.foodbankcenc.org.

[38] Seminar on Data and Information Management. http://www.iwi.uni-hannover.de/lv/seminar_ss05/bartke/geomet.htm.

[39] ParaView. http://www.paraview.org/.

[40] Adam Perer and Ben Shneiderman. Systematic yet flexible discovery: guiding domain experts through exploratory data analysis. In *Proceedings of the 13th international conference on Intelligent user interfaces*, IUI '08, pages 109–118, New York, NY, USA, 2008. ACM.

[41] Raquel M. Pillat, Eliane R. A. Valiati, and Carla M. D. S. Freitas. Experimental study on evaluation of multidimensional information visualization techniques. In *Proceedings of the 2005 Latin American conference on Human-computer interaction*, CLIHC '05, pages 20–30, New York, NY, USA, 2005. ACM.

[42] PostGIS. http://postgis.refractions.net/.

[43] PostgreSQL. http://www.postgresql.org/about/.

[44] Prefuse. http://prefuse.org/.

[45] Processing. http://processing.org/.

[46] Prototype. http://prototypejs.org/.

[47] Zou Qingsong, Kwoh Chee Keong, and Ng Wan Sing. Interactive surgical planning using context based volume visualization techniques. In *Proceedings of the International Workshop on Medical Imaging and Augmented Reality (MIAR '01)*, MIAR '01, pages 21–, Washington, DC, USA, 2001. IEEE Computer Society.

[48] R. `http://www.r-project.org/`.

[49] rApache. `http://rapache.net/`.

[50] Raphael. `http://raphaeljs.com/`.

[51] IBM Research and IBM Cognos software group. `http://www-958.ibm.com/software/analytics/manyeyes/`.

[52] John Risch, Anne Kao, Stephen R. Poteet, and Y. J. Wu. Visual data mining. chapter Text Visualization for Visual Text Analytics, pages 154–171. Springer-Verlag, Berlin, Heidelberg, 2008.

[53] Timo Ropinski, Steffen Oeltze, and Bernhard Preim. Survey of glyph-based visualization techniques for spatial multivariate medical data. *Computers & Graphics*, 35(2):392 – 401, 2011. Visual Computing in Biology and Medicine.

[54] N Shetty, A Chaudhary, D Coming, W R. Sherman, P O'Leary, E T. Whiting, and S Su. Immersive paraview: A community-based, immersive, universal scientific visualization application. In *Proceedings of the 2011 IEEE Virtual Reality Conference*, VR '11, pages 239–240, Washington, DC, USA, 2011. IEEE Computer Society.

[55] Artur Silic, Annie Morin, Jean-Hugues Chauchat, and Bojana Dalbelo Basic. Visualization of temporal text collections based on correspondence analysis. *Expert Syst. Appl.*, 39(15):12143–12157, November 2012.

[56] Tableau Software. `http://www.tableausoftware.com/public/`.

[57] SpryMedia. `http://www.datatables.net/`.

[58] Dan Sunday. `http://geomalgorithms.com`.

[59] Roger C. Tam, Christopher G. Healey, Borys Flak, and Peter Cahoon. Volume rendering of abdominal aortic aneurysms. In *Proceedings of the 8th conference on Visualization '97*, VIS '97, pages 43–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[60] Thinkmap. `http://www.visualthesaurus.com/`.

[61] Google Chart Tools. `https://developers.google.com/chart/interactive/docs/index`.

[62] Siyavula Uploaders. http://cnx.org/content/m29760/latest/.

[63] VisIt. https://wci.llnl.gov/codes/visit/about.html.

[64] VTK. http://www.vtk.org/.

[65] World Wide Web Consortium (W3C). http://www.w3.org/XML.

[66] John A. Wass. Spotfire: A visual experience, May 2010.

[67] Zhu Wei-ping, Li Ming-xin, and Chen Huan. Using mongodb to implement textbook management system instead of mysql. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pages 303 –305, may 2011.

[68] Thomas Wischgoll. http://avida.cs.wright.edu/courses/CS399/CS399_4.pdf.

[69] Thomas Wischgoll. http://avida.cs.wright.edu/courses/CS399/CS399_5.pdf.

[70] Yahoo! http://yuilibrary.com/.

[71] Ranran Yang, Zhanqiang Chang, and Tengfei Xue. 3d terrain visualization for mountain taishan. In *Spatial Data Mining and Geographical Knowledge Services (ICSDM), 2011 IEEE International Conference on*, pages 285 –290, 29 2011-july 1 2011.

[72] Xiaoqiang Zheng and Alex Pang. Volume deformation for tensor visualization. In *Proceedings of the conference on Visualization '02*, VIS '02, pages 379–386, Washington, DC, USA, 2002. IEEE Computer Society.