A Three-Dimensional Geographic and Storm Surge Data Integration System for Evacuation Planning

Jairo Pava[1], Fausto Fleites[1], Fang Ruan[1], Kasturi Chatterjee[1], Shu-Ching Chen[1], Keqi Zhang[2]
[1]*Distributed Multimedia Information Systems Laboratory*
[2]*International Hurricane Research Center*
Florida International University, Miami, FL 33199
[1]*{jpava001, fflei001, ruanf, kchat001, chens}@cs.fiu.edu*, [2]*zhangk@fiu.edu*

Abstract

The rise of offshore water caused by the high winds of a low pressure weather system, or storm surge, is a hurricane's greatest threat to human life. As weather forecasters struggle to enable coastal residents to make timely evacuation decisions, the need arises for more visually compelling and interactive storm surge visualization tools. This paper presents an interactive and three-dimensional storm surge visualization system. It integrates road, topographic, and building data to construct accurate three-dimensional models of major cities in the State of Florida. Storm surge data are then used to construct a three-dimensional ocean positioned over the terrain models. Ambient details such as wind, vegetation, ocean waves, and traffic are animated based on up-to-date wind and storm surge data. Videos of the storm surge visualizations are recorded and made available to coastal residents through a web-interface. The three-dimensional visualization of geographic and storm surge data provides a more visually compelling representation of the potential effects of storm surge than traditional two-dimensional models and is more capable to enable coastal residents to make potentially life-saving evacuation decisions.

Keywords: Information Integration, Data Visualization, Geographic Information Systems, Storm Surge

1. Introduction

Hurricanes are the most destructive natural hazards to threaten the United States East and Gulf coasts. The greatest threat to human life and property by a hurricane is storm surge. Historically, storm surge has led to 90% of fatalities caused by hurricanes. Salt water flooding is also a major cause of damage to buildings and infrastructure [5][7].

As low pressure weather systems, such as hurricanes, approach a coastal area, storm surge models are used to estimate the potential storm surge at the projected landfall area. One such type of model is the Coastal and Estuarine Storm Tide (CEST) model developed by the International Hurricane Research Center (IHRC) [22][25]. The model takes into consideration the expected tide at landfall and atmospheric pressure and wind of the weather system. It also takes into consideration major coastal topographic features such as coastal ridges and barrier islands. Based on storm surge models, coastal residents are warned and advised on evacuations through television, Internet, radio, and newspapers. Coastal residents, however, have been unable to relate the two-dimensional evacuation maps broadcast over media outlets to their three-dimensional life experiences. Consequently, emergency planners have struggled to avoid under and over-evacuation of coastal areas.

To address the issue of applying information and knowledge for enhancing timely evacuation decision-making, we propose a three-dimensional storm surge visualization system that integrates Light Detection and Ranging (LiDAR) measurements, storm surge data, and road data in Digital Line Graphics (DLGs) in the State of Florida to effectively aid coastal residents as they prepare for a storm. With the term three-dimensional, we refer to a computer-generated virtual environment that may be interacted with and viewed within the spatial dimensions of depth, height, and width. Coastal residents will better relate to three-dimensional animations of their homes or businesses under storm surge conditions and instantly be more disposed to regard official evacuation notices. The work in [23] presents a storm surge visualization system to display the effects of a rising storm surge on vehicles during a hypothetical hurricane. However, it is highly scripted and the individual components used to animate the storm surge, wind, traffic, and rain have to be carefully configured before every initialization. Therefore, it is not practical to use such a time-consuming system when coastal residents must make time-sensitive evacuation decisions hours before storm impact. Furthermore, the system does not have support for any formal storm surge model and can therefore not be used to simulate official storm surge data.

Our proposed system will extend the work in [23]. It will utilize data from the CEST model to automatically configure and update the storm surge visualization system in real-time as updated storm surge data become available. Real-time refers to the latest official storm surge projections which are typically updated every three to six hours as a hurricane approaches landfall. In this manner, coastal residents will be better informed when making potentially life-saving evacuation decisions. Additionally, researchers will now have the ability to use our system with historical data to further study the effects of storm surge and help mitigate its damage in the future. We have also developed a web-based interface to overcome the hardware limitations imposed by [23] so that all coastal residents and emergency planners, regardless of their available computational power, have access to our system's storm surge visualizations.

The main contributions of this paper are as follows: (1) it provides techniques for a synergetic integration of LiDAR and storm surge data; (2) it presents a novel methodology to accurately visualize real-time and historical hurricane storm surge scenarios in a three-dimensional environment; and (3) it elaborates on a web-based integration of two and three-dimensional storm surge visualizations for coastal residents and emergency planners.

The rest of this paper is organized as follows. The next section discusses the system architecture of our visualization system. Section 3 describes how our system procedurally generates digital terrain models (DTMs) and build footprints from LiDAR measurements and roads from DLG data. Section 4 describes how the output from the CEST model is used to animate the visualization. Section 5 describes a web-based interface for coastal residents and emergency planners to view storm surge visualizations. Section 6 discusses related work. Sections 7 and 8 present concluding remarks and acknowledgements.

2. System Architecture

The proposed system utilizes and extends the facilities in the open-source Virtual Terrain Project (VTP) [13]. The VTP is a set of two programs, VTBuilder and Enviro, which are used to construct and render three-dimensional visualizations of geographic data. VTBuilder provides an interface to construct terrains, structures, and roads. Enviro provides a highly interactive three-dimensional runtime environment using OpenGL [10]. We have extended both programs to integrate large CEST model data sets, LiDAR measurements, and DLGs for roads into a compelling three-dimensional visualization of storm surge data.

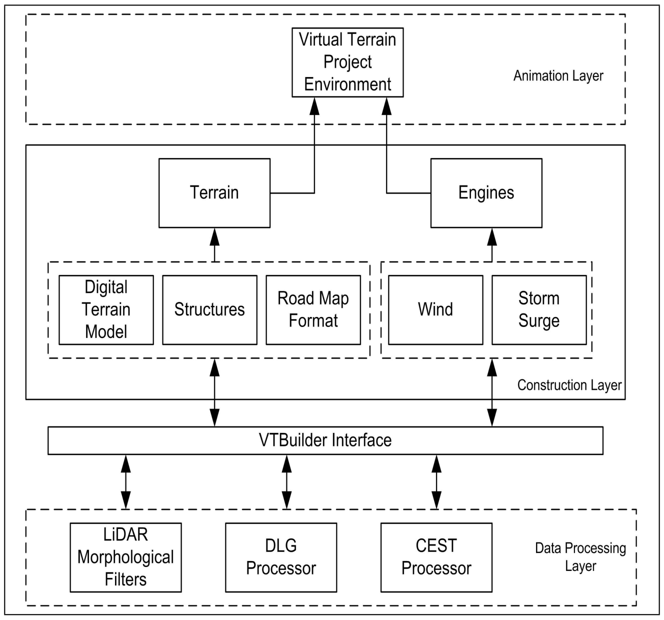

Figure 1. System Architecture

Figure 1 presents a high-level four-tier system architecture used to integrate geographic and storm surge data for interactive visualization. The data processing layer provides the data management facilities that allow for conversion of LiDAR, DLG, and CEST data into formats that serve as accessible input data to VTBuilder. Processed input data is used in the construction layer to construct three-dimensional representations of terrain, structures, and roads. These data are then integrated to construct a final digital representation of the terrain with roads and buildings. Wind and storm surge data, obtained from the CEST model, are used in the construction layer to drive graphics animation engines for wind, storm surge, vegetation, rain, traffic, and ocean waves. Access to the VTBuilder interface is bi-directional so that a user may modify processed data after it has been constructed into three-dimensional form. The DTM, buildings, roads, and graphics animation engines are then integrated and passed on as inputs to the VTP's Enviro for final rendering in the animation layer. This layer provides an interactive three-dimensional navigation of the digital terrain along with animated storm surge, rain, lighting, cloud, vegetation, traffic, and sun visualizations.

3. Terrain Data Integration

In this section we provide background information on LiDAR and DLG data and how they are used to accurately represent topography, buildings, and roads in our system.

We then discuss how these data are integrated into a standard format usable by the VTP for three-dimensional visualization.

3.1. LiDAR

LiDAR is a remote sensing system used to collect topographic data. The National Oceanic and Atmospheric Administration (NOAA) collects LiDAR data with aircraft-mounted lasers that are capable of recording elevation measurements at a rate of 2,000 to 5,000 pulses per second with a vertical precision of six inches. LiDAR data points are stored as XYZ data points in text files where x is longitude, y is latitude, and z is the elevation at that point. The laser-scanned data include topographical measurements as well as non-ground objects such as cars, buildings, and vegetation. The data must, therefore, be processed into separate measurements for ground and non-ground objects to be usable for three-dimensional animation.

3.2. Digital Terrain Model

To construct a DTM from LiDAR data, measurements from ground and non-ground features have to be identified and categorized. The methodology used to remove non-ground measurements from LiDAR data is the progressive morphological filter investigated by Zhang et al. in [24]. The filter is based on dilation and erosion fundamentals of set theory and has the advantage of filtering out non-ground objects without prior knowledge of the size and elevation of the terrain. Additionally, the morphological filters automatically derive elevations for gaps in the LiDAR data using nearest neighborhood interpolation. The resulting output from the morphological filters is converted to the Binary Terrain (BT) [2] format with VTBuilder. A BT file is an elevation grid consisting of elevation values at specified geo-coordinates. BT files are used by Enviro to render three-dimensional representations of DTMs.

In an effort to model the geographic extents of the State of Florida, DTMs of nine cities have been created using LiDAR data. They are Pensacola, Jacksonville, Tampa, St. Petersburg, West Palm Beach, Ft. Lauderdale, Miami Beach, Key Biscayne, and Key West.

3.3. Automatic Building Construction

Extending upon the morphological filter in [24], we use the framework in [26] to automatically construct buildings from LiDAR measurements. Ground and non-ground LiDAR measurements are separated using the progressive morphological filter. Building measurements are then identified using a region-growing algorithm based on plane-fitting techniques. Raw footprints for building measurements are identified by connecting boundary points. These raw footprints are then automatically refined to remove noise caused by irregularly spaced LiDAR measurements. The heights of the buildings are identified by averaging the elevation differences between building measurements and the DTM constructed from non-ground measurements.

```
<StructureCollection
 xmlns:gml="http://www.opengis.net/gml">
 <Building>
   <Level FloorHeight="3.0"
     StoryCount="1">
       <Footprint>
           <gml:coordinates>
               587225.873,2851049.22
               587206.933,2851075.11
               587200.844,2851059.76
               587223.588,2851034.80
           </gml:coordinates>
       </Footprint>
       <Edge Material="Siding"
         Color="ffffff">
           <EdgeElement Type="Wall"/>
       </Edge>
   </Level>
 </Building>
</StructureCollection>
```

Figure 2. VTST Structure File

After identifying building footprints, VTBuilder is used to construct a Virtual Terrain Structure (VTST) [14] file based on the extensible markup language (XML) format. VTST is readable by the VTP and allows for systematic creation of 3D building models based on the footprints. VTST is built upon the definition for the OpenGIS Geography Markup Language (GML) Implementation Specification, version 2.1.2 [9]. It is the standard for encoding geographic data in XML.

Figure 2 presents a sample building footprint in the VTST format. Building footprint points are specified in counter-clockwise order using the VTP's own two-dimensional coordinate system which corresponds to the geo-coordinates in the LiDAR data. The building's floor height is specified in meters, and a building may have more than one floor with respect to the story count attribute. The footprint vertex with the lowest elevation becomes the height of the base of the building. Furthermore, buildings may be systematically textured by adding the Edge element with the appropriate attributes for material and color.

There exist advantages to using LiDAR data to construct building models rather than 3D model creation software as proposed in [23]. By using geo-referenced LiDAR data

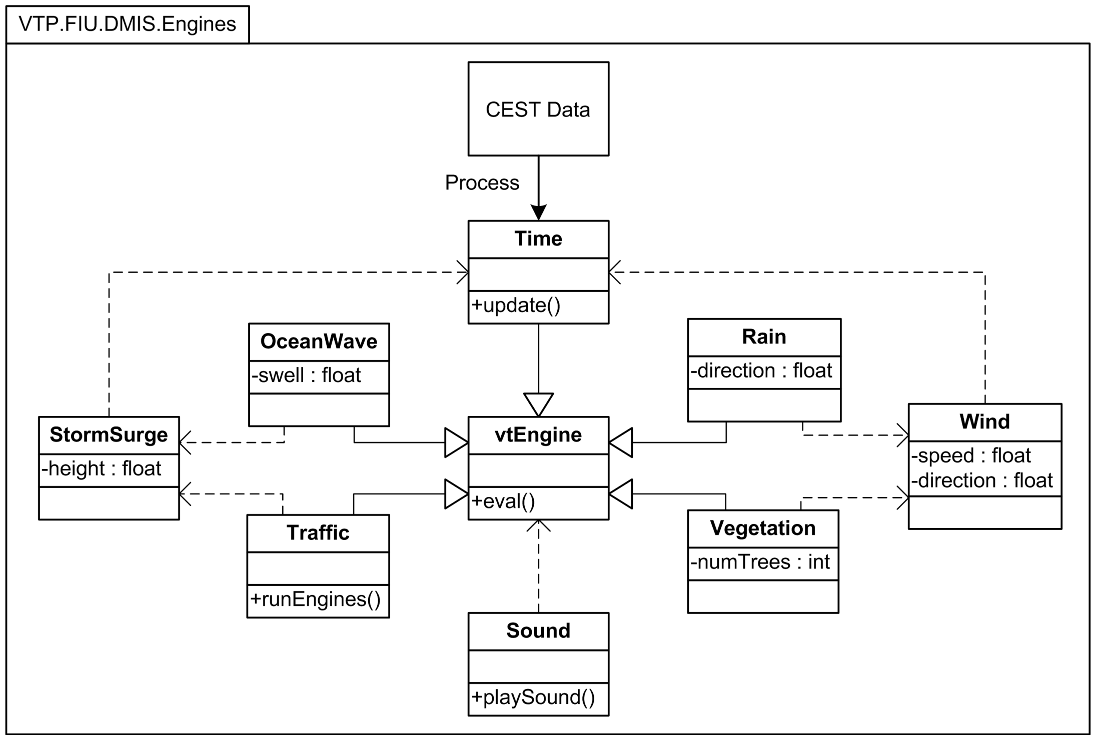

Figure 3. System engines and modules

points, a building is placed at its exact geo-coordinates on the DTM and with accurate distance between its neighboring buildings. We are able to do this within minutes of processing LiDAR data whereas it typically takes two to three hours, on average, to manually design one building model. Most importantly, we can be certain that building heights are accurate and the relative storm surge height in the visualization is precise. However, while the VTST files allow us to procedurally texture the generated building models, we must manually texture landmark buildings to look more realistic.

3.4. Road Distribution

United States Geological Survey (USGS) DLG data are digital vector representations of cartographic information. The data are publicly available for download from the official USGS website [12]. Since the road data captured in DLG comes from scanned cartography, it does not contain road widths, names of streets, or traffic direction. However, the precise geo-coordinates of the DLG files as well as publicly available road map data [4] allow our system to render roads with accurate placements and dimensions. VTBuilder is then used to convert the data from DLG into Road Map Format (RMF) [11], a binary file format that stores the extents of the DLG dataset and road coordinates and intersections. Enviro uses the RMF file to appropriately render the roads over the terrain.

4. CEST Integration

In this section we describe how the IHRC's CEST model is used to drive all of the graphics animation engines and modules in the VTP to accurately and realistically render three-dimensional storm surge scenarios.

Figure 3 presents a minimal class diagram of all the engines that utilize CEST model data for animation. The vtEngine is an abstract class provided by the VTP that coordinates the behavior of every animation within visualization. Its abstract eval() function is invoked once per frame and must be implemented by every class that inherits from vtEngine with code that renders some change to the animation. All of the engines that inherit or are otherwise associated with vtEngine are described in the following sections.

4.1. Time Engine

The Time class implements the time engine to update the wind and storm surge modules. It keeps track of the time in terms of days, hours, minutes, and seconds. It can also be set to begin at specific times upon initialization for

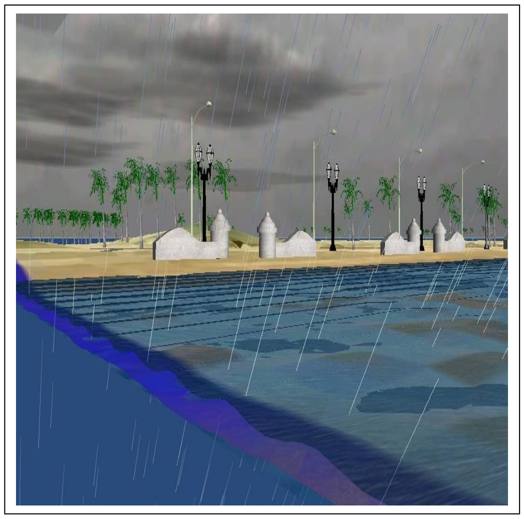

Figure 4. Ocean wave simulation

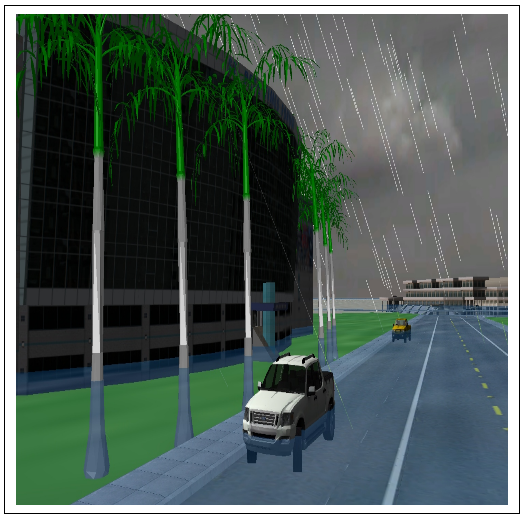

Figure 5. Vehicles respond to storm surge

simulation of historical or projected storms and can elapse at accelerated rates. At every invocation of its eval() function, the time engine updates the current time and reads in the wind and storm surge data corresponding to the same time interval in the supplied CEST data. At the completion of every interval in the simulation, the time engine updates the storm surge and wind modules with new CEST data.

4.2. Wind Module

At every update, the wind module, implemented by the Wind class, receives updated CEST model wind data from the time engine and correspondingly updates its current wind direction and speed by calculating an average of all the wind measurements within a one mile radius from the currently simulated location. It does not need to keep track of the next set of wind data to read at the following update as this is managed by the time engine. It does not inherit from the vtEngine class since it only needs to be updated when a new time interval has been approached by the time engine. Wind direction is measured in degrees and wind speed is measured in miles per hour. To maintain a synchronized and fluid animation with respect to wind, the wind module is invoked once per frame by the vegetation and rain engines.

4.3. Vegetation and Rain Engines

The Vegetation class implements the vegetation engine to manage all of the trees in the animation. It uses an implementation of vertex weighting [20] for animation of three-dimensional trees. At every frame in the animation, the vegetation engine performs rotation, bend, and rupture of individual tree branches based on wind speed and direction data. Tree animation is an integral component of ambient details that adds to the experience of the visualization.

Similarly, the Rain class implements the rain engine to render the direction and speed of rain droplets based on wind data. While the CEST model does not predict precipitation, other sources may be used to automatically gather and incorporate this data into the visualization [6].

4.4. Storm Surge Module and Ocean Wave Engine

The storm surge module, implemented by the StormSurge class, receives updated CEST model data from the time engine and updates the current storm surge height by calculating an average of all the storm surge measurements within a one mile radius from the currently simulated location. The storm surge module is invoked by the ocean wave engine once per frame to render the varying height and waves of the ocean.

The ocean is modeled by animating a mesh using Fournier's model for ocean waves [18]. This model animates water particles in circular or elliptical stationary orbits and takes into account the surface of the ocean's floor for breaking waves on the shore. Specifically, each water particle describes a circle around its rest position (x_0, y_0, z_0), and the particle's motion around the circle is given by the following parametric equations in x_0 for a given t and a constant z_0:

$$x = x_0 + rsin(kx_0 - \omega t)$$
$$z = z_0 - rcos(kx_0 - \omega t)$$

The XZ plane is the horizontal plane, and the height of a wave is given on the Z axis. The above equations describe the curve generated by a point at a distance r from the center of a circle of radius $\frac{1}{k}$; the circle rolls over a line at distance $\frac{1}{k}$ under the X axis [18]. For $t = 0$ and $z_0 = 0$, the equations are

$$x = -\frac{\alpha}{k} - r sin(\alpha)$$
$$z = -r cos(\alpha)$$

where $\alpha = -kx_0$, the height of the wave is $H = 2r$, the wavelength is $L = \frac{2\pi}{k}$, the period is $T = \frac{2\pi}{\omega}$, the phase speed is $c = \frac{L}{T} = \frac{\omega}{k}$, and the phase is $\phi = kx_0 - \omega t$ - assuming a phase of 0 for x_0. In deep ocean, the period and wavelength are related by $L = \frac{gT^2}{2\pi}$ [18].

The surface of the ocean's floor is accounted for by including the depth h at the point (x_0, y_0, z_0) in a cumulative way, meaning that the phase delay introduced by the depth effect is carried over from deep ocean to the shore. The wave number k (the reciprocal of the wavelength) is a function of h, h is a function of x_0, and the phase is given by the equation:

$$\phi = -\omega t + \int_0^{x_0} k(x)dx \text{ where } k(x) = \frac{k_\infty}{\sqrt{tanh(k_\infty h(x))}}$$

The term k_∞ is the wave number at deep ocean (infinite depth) and is calculated by $k_\infty = k tanh(kh)$. An approximation to the above equation is given by [18]:

$$\phi = -\omega t + \sum_0^{x_0} \frac{k_\infty}{\sqrt{tanh(k_\infty h(x))}} \Delta x$$

The angular frequency ω and the wave number k utilized to calculate k_∞ are given as input parameters in our simulation, and the time t is obtained from the temporal succession of frames of the wave engine. The waves begin their motion at deep ocean, where the depth is an arbitrary large number $MAXDEPTH$, and move towards the shore, where $h(x)$ is simulated as an increasing slope from $MAXDEPTH$ to zero depth on the shore. Figure 4 shows our water animation using the model just described.

4.5. Traffic Engine

The traffic system engine developed in [19] is used to render traffic on the roads placed on the terrain. The traffic engine keeps track of all the vehicles in the animation and updates the position of vehicles at every frame while making sure that they follow road directions (stop lights, lanes, road direction, etc.) and avoid collisions with each other. Furthermore, the traffic engine's dependency to the storm surge module enables it to respond to changes in storm surge and correspondingly force all vehicles to slow down or stop if the storm surge rises beyond predetermined thresholds. Figure 5 presents a screenshot of stalled vehicles due to high storm surge.

4.6. Sound

OpenAL is used to implement sound in our system. It is a free cross-platform audio Application Programming Interface (API) used in many games and simulation software [8]. OpenAL is designed for efficient three-dimensional audio. That is, any engine in the animation may specify the velocity, position, direction, and intensity of a sound in a three-dimensional space. This provides a more realistic auditory component in the animation for wind, rain, vehicles, lightning, and vegetation. For example, as wind speed increases, the intensity of the professionally-recorded sound also increases. Furthermore, as one navigates through the visualization, the direction and intensity of sound changes with respect to the current location in the terrain and the sound sources. The Sound class is a singleton and therefore vtEngine has a reference to only one instance. This allows it to coordinate sounds from all the engines through its playSound() function.

5. Web-based Interface

To make the storm surge visualizations accessible to coastal residents, we have designed a web-based interface. It is based on the Google Maps API [3] and uses HTML, JSP, Ajax, JavaScript, and Flash web technologies. Figure 6 presents a screen capture of the web interface. For clarification, areas in the web interface discussed below are labeled with a number enclosed in a box.

An interactive map of the State of Florida consumes major part of the user's screen. To the right of the screen, a sliding bar menu (label 1) contains an address locator which prompts the user to enter the address of a storm surge visualization he or she is interested in seeing. Upon having an address submitted, the web interface queries a database of user-contributed storm surge multimedia within a twenty mile radius. It then uses Ajax to seamlessly display the retrieved images or videos without refreshing the page (label 2). Simultaneously, the map on the screen centers on the address submitted, and a slightly transparent and color-coded image (label 3) is superimposed over the map. The colors of the image reflect the storm surge heights provided by the latest CEST model data. If a visualization of the storm surge is available at the location requested, a video icon (label 4) appears on the map. Clicking on the icon reveals a window, presented in Figure 7, with a video of the storm surge visualization. Videos are presented in Flash video format, have sound, and last for sixty seconds. Users have the option to set the size of the video to fit their screens. The videos are automatically updated every three to six hours as updated storm surge data becomes available.

By presenting the visualizations of the storm surge scenarios as videos, coastal residents are able to view the visu-

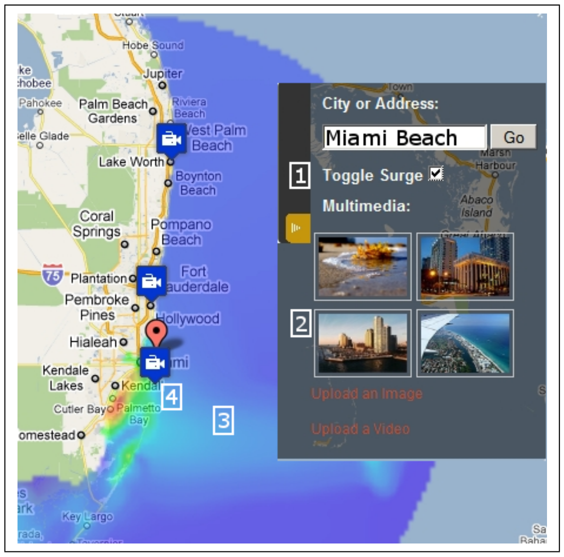

Figure 6. Web Interface

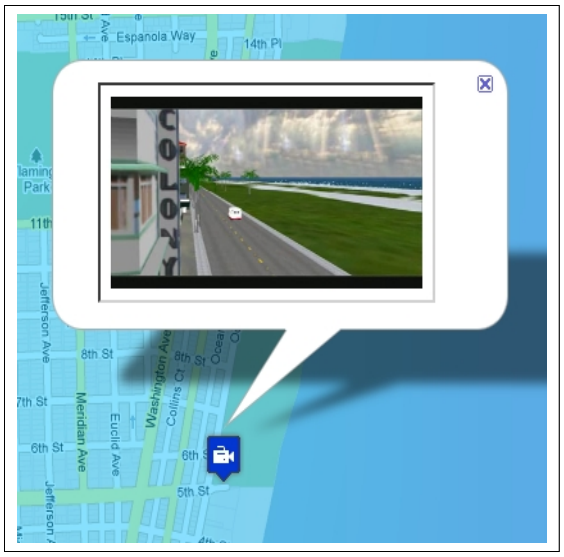

Figure 7. Storm Surge Video

alizations regardless of the performance capacities of their personal computers. Additionally, users can access the video through any mobile device with a Flash-supporting web browser. This removes the limiting requirement of needing a personal computer and access to a power source to access the visualizations.

6. Related Work

There has been great effort in the research community to provide alternative ways to visualize geographic and storm surge data. However, most of the work that focuses on storm surge visualization relies heavily on manual configuration and is therefore too slow to be effective for time-sensitive evacuation planning. In contrast, our approach uses an automated solution to automatically integrate geographic and storm surge data and systematically generate storm surge visualizations.

The work presented by Venkataraman et al. [15] is most closely related to our work. The proposed work integrates computerized forecast models for hurricane wind, temperature, and storm surge and LiDAR and GIS for three-dimensional terrain views. The data are integrated into a unifying data format, Hierarchical Data Format V.5, and rendered using the Amira visualization framework [1]. Amira, however, does not have Level-of-Detail (LOD) algorithms to handle large LiDAR data sets and cannot render interactive and animated visualizations like the VTP used in our approach.

Webster et al. [21] use topographic LiDAR measurements to construct a high-resolution digital elevation model (DEM) for Charlottetown, Prince Edward Island, Canada. Storm surge data are represented as a two-dimensional shaded-relief image superimposed over the DEM in a GIS rendering system. However, in this paper we focus on producing three-dimensional visualizations of storm surge scenarios data that are more likely to persuade coastal residents to follow official evacuation notices. Furthermore, we provide a web-based interface that enables coastal residents to view storm surge scenarios for a wider set of city locations.

Flaxman et al. [17] uses the VTP to visualize alternative futures for large cities and regions. It discusses a case study where the city of Hangzhou, China, was modeled using existing GIS data for buildings, terrain, rivers, and roads. The city models were used to help city planners identify areas to place strategic public investments in transportation, infrastructure, and civic buildings. Our approach differs from [17] in that we utilize LiDAR data to automatically construct building models and terrains for storm surge planning. By using LiDAR data, we can procedurally construct accurate city models with considerably less manual effort. However, our system can benefit from the methods used in [17] to evaluate our system's ability to operate in an urban planning environment.

7. Conclusions and Future Work

In this paper we present a three-dimensional storm surge visualization system. LiDAR data were used to extract buildings and terrain topography measurements to construct digital terrain models. Roads were accurately placed on the terrains using USGS DLG data. Finally, CEST model data were used to construct three-dimensional visualizations of storm surge data on the terrains. Additionally, CEST model data were used to drive sound and wind, ocean wave, rain, vegetation, and traffic animations within the visualization. An accessible and portable web-based interface for the storm surge visualizations based on the Google Maps API was also presented. Videos of storm surge visualizations are updated based on the the most up-to-date storm surge projections and automatically uploaded to the web-interface. By integrating LiDAR, DLG, and CEST model data, we have successfully developed a three-dimensional data visualization system that may be used as a public relations tool to better inform coastal residents of the potential storm surge in their immediate area and enable them to make potentially life-saving decisions. Furthermore, decision makers are enabled to perform optimal urban planning and mitigate flood damage by analyzing potential storm surge scenarios.

Future work calls for: (1) using more extensive LiDAR data sets to create storm surge visualizations for the entire State of Florida; (2) investigating how the community reacts to the storm surge visualizations and evaluating what is the most useful information to coastal residents and public decision makers; and (3) exploring methods of enabling users to interact with the storm surge visualization system through the web-based interface.

Acknowledgements

This project was supported in part by a grant from NOAA.

References

[1] Amira, http://www.amira.com/ (Apr. 2010).

[2] BT Format, http://vterrain.org/implementation/formats/bt.html (Apr. 2010).

[3] Google Maps API, http://code.google.com/apis/maps (Apr. 2010).

[4] Google Maps, http://maps.google.com (Apr. 2010).

[5] IHRC, http://www.ihrc.fiu.edu/about_us/hurricane_hazards (Apr. 2010).

[6] National Digital Forecast Database Simple Object Access Protocol Web Service, http://www.nws.noaa.gov/xml (Apr. 2010).

[7] NOAA, http://www.nhc.noaa.gov/haw2/english/storm_surge.shtml (Apr. 2010).

[8] OpenAL, http://connect.creativelabs.com (Apr. 2010).

[9] OpenGIS, http://www.opengeospatial.org (Apr. 2010).

[10] OpenGL, http://www.opengl.org (Apr. 2010).

[11] RMF, http://vterrain.org/doc/roads.html (Apr. 2010).

[12] United States Geological Survey, http://data.geocomm.com (Apr. 2010).

[13] Virtual Terrain Project, http://vterrain.org (Apr. 2010).

[14] VTST Format, http://vterrain.org/implementation/formats/vtst.html (Apr. 2010).

[15] W. Benger, A. L. S. Venkataraman, G. Allen, S. D. Beck, M. Brodowicz, J. Maclaren, and E. Seidel. Visualizing katrina - merging computer simulations with observations. In *Springer Verlags Lecture Notes in Computer Science Series (in press)*.

[16] S.-C. Chen, K. Zhang, and M. Chen. A real-time 3d animation environment for storm surge. In *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo*, pages 705–708, Washington, DC, USA, 2003. IEEE Computer Society.

[17] M. Flaxman. Using the virtual terrain project to plan real cities: alternative futures for hangzhou, china. In *ACM SIGGRAPH*, pages 340–350, San Antonio, TX, USA, 2002.

[18] A. Fournier and W. T. Reeves. A simple model of ocean waves. In *SIGGRAPH*, pages 75–84, 1986.

[19] Y. Li, K. Chatterjee, S.-C. Chen, and K. Zhang. A 3-d traffic animation system with storm surge response. In *ISM '09: Proceedings of the 2009 11th IEEE International Symposium on Multimedia*, pages 257–262, Washington, DC, USA, 2009. IEEE Computer Society.

[20] K. Saleem, S.-C. Chen, and K. Zhang. Animating tree branch breaking and flying effects for a 3d interactive visualization system for hurricanes and storm surge flooding. In *ISMW '07: Proceedings of the Ninth IEEE International Symposium on Multimedia Workshops*, pages 335–341, Washington, DC, USA, 2007. IEEE Computer Society.

[21] S. D. T.L. Webster, D.L. Forbes and R. Shreenan. Using topographic lidar to map flood risk from storm-surge events for charlottetown. In *Canadian Journal of Remote Sensing*, pages 64–76, 2004.

[22] C. Xiao, K. Zhang, and J. Shen. A three-dimensional coastal and estuarine storm tide model. *Journal of Coastal Research*, page 20, 2006.

[23] K. Zhang, S.-C. Chen, P. Singh, K. Saleem, and N. Zhao. A 3d visualization system for hurricane storm-surge flooding. *IEEE Comput. Graph. Appl.*, 26(1):18–25, 2006.

[24] K. Zhang, S. ching Chen, D. Whitman, M. ling Shyu, J. Yan, C. Zhang, and S. Member. A progressive morphological filter for removing nonground measurements from airborne lidar data. *IEEE Transactions on Geoscience and Remote Sensing*, 41:872–882, 2003.

[25] K. Zhang, C. Xiao, and J. Shen. Comparison of the cest and slosh models for storm surge flooding. *Journal of Coastal Research*, 24:489–499, 2008.

[26] K. Zhang, J. Yan, and S. Chen. Automatic construction of building footprints from airborne lidar data. *IEEE Transactions on Geoscience and Remote Sensing*, 44(9):2523–2533, September 2006.