A 3-D Traffic Animation System with Storm Surge Response

Yudan Li, Kasturi Chatterjee, Shu-Ching Chen
Distributed Multimedia Information Systems Laboratory
School of Computing and Information Sciences
Florida International University, Miami, FL, USA
{yli011,kchat001,chens}@cs.fiu.edu

Keqi Zhang
International Hurricane Research Center
Florida International University
Miami, FL, USA
zhangk@fiu.edu

***Abstract*—This paper presents a 3D traffic animation system that provides features of vehicle interaction and storm surge response to emulate realistic scenarios in a hurricane affected area. The proposed system constructs road objects based on a series of line segments. Vehicles on the roads are animated by keeping them coherent with the direction of the road segment and adjusting their speeds with respect to one another. The vehicle speed is controlled based upon a collision-circumventing policy and they automatically respond to flood water caused by a storm surge. The system can automatically plant lamp posts along the roads, which also respond to surge flooding. An implementation conducted with the 3D scenes on a couple of south Florida's surge-susceptible areas demonstrates the system's excellence in animating real traffic scenes and impacts of storm surge on coastal regions.**

***Keywords*-animation; 3D rendering; traffic system; storm surge**

I. Introduction

Three-dimensional traffic animation is of great significance in the field of disaster impact prediction and evaluation. This has grown over years to be an important area of research. Institutions related to weather forecast, disaster control, civilian safety protection as well as insurance companies have benefited from the successful implementation of such techniques. A traffic system plays a principal role in a disaster animation scene because (a) it is a component in which human beings are directly involved via their vehicles and thus any disruption in it affects human life closely; (b) it provides prior information about safe routes for evacuation when the area is affected by a disaster. Although traffic animation is also applied in other situations like urban planning or transportation management, these areas usually do not cover all the requirements of a typical disaster management system. Urban planning [1] usually targets on a high definition demonstration and overlooks interaction details while transportation management [2] focuses on vehicle interaction but may overlook visual effects.

The 3D Visualization and Animation of Storm Impacts [3] [4] is a typical example of a disaster management project. The project provides predictive information about storm surge impacts with highly realistic 3D scenes. The coastal regions of south Florida are highly susceptible to effects of hurricanes. One of the major effects of hurricanes is storm surge flooding which affects the traffic systems of these regions severely. Since, safe evacuation is an important issue for hurricane affected areas, the effects of storm surge on the traffic systems is crucial to the common people as well as to the officials responsible for their safety. This project aims to provide a public training medium by providing realistic animation of storm surge effects on the traffic systems. It would help the people to evaluate a hurricane scenario and plan effectively in case of an emergency. The system has three main requirements: (a) high definition and accuracy; (b) realistic interaction between various object pairs like vehicle-vehicle and vehicle-flood water; (c) low computation overhead. An ideal animated disaster management system should cover the interaction between vehicles and disaster impacts. At the same time it should keep the principle of vehicle animation simple but effective so that the computational complexity is low. In this paper, we present a novel traffic animation system, based on road segmentation, which emulates storm surge impacts on the traffic systems in coastal regions of south Florida. Simpler traffic representations are used with realistic effects to lower computational overheads.

The rest of the paper is organized as follows: Section II reviews the existing works. Section III describes the initial setup of road segment objects. Section IV describes vehicle setup and animation principles. Section V describes the interaction between the vehicles and their response to storm surge. This is followed by Section VI which demonstrates the effect of the proposed 3D traffic system on a couple of flood-prone south Florida locations followed by a brief conclusion in Section VII.

II. Literature Review

A traffic animation system basically consists of two parts: road representation and vehicle movement. Roads can be represented in a number of ways. A list of vehicle-centered 3D points denoting left and right edges were used to represent the road in [5]. A sequence of vanishing points can be used to represent the 3D shape of a curved road as in [6] [7]. A differential geometry, decoupled for horizontal and vertical curvature, is another solution for road representation [8]. Alternatively, the road can also be modeled as features which form concentric circular arcs lying on a flat ground plane [9]. The two-edge-based method [5] only provides information about where the vehicle can't go. Thus a center-line based representation is more preferable in explicitly indicating a specific route for the vehicle to follow. The vanishing point-based methods [6] [7] involve a vanishing point-road direction conversion, which is an extra overhead. The method, working on differential geometry [8], is a type of continuous representation, requiring complicated computations for vehicle movement. Also, the arcs-based representation [9] needs heavier calculation while it does not enhance the visual effects rendered for typical systems as much.

There are also a number of methods for animating vehicle movement. However, the method to be applied is dependent on the kind of road representation adopted. In a simple case, in which a road is divided into cells of equal lengths [10], a car tries to advance to the next cell at each time state. If the next cell is occupied, the car stops. Such a mechanism requires hard coding of all the cells composing the road and lacks flexibility in simulating the car's in-cell status. For vehicle movements under other types of road representations mentioned above, a set of position updating calculations need to be conducted [8] [5] [6] [7]. The computation complexity, involved in implementing these systems, is an issue of consideration for generating optimum rendering effect.

The study of vehicle interactions has long been conducted by researchers in the field of smart vehicle navigation [11] [12]. Logic throttle and brake controller were designed for speed control [11]. Besides, vehicle operation can be defined into three categories [12]: collision avoidance, detection scheme and multiple vehicle operation. Although there's some similarity in the concept of speed control between the domains of smart vehicle navigation and 3D animation, further research need to be conducted to cater to the specific requirements of 3D animation systems. Components such as frame-based operations, reducing computation overhead and introducing application-dependent effects like storm surge, are some of them.

In this paper, we present an animated traffic system that implements an efficient road representation. Additionally, it renders realistic vehicle movement while incurring low computational overhead. Furthermore, it responds to storm surge flooding, thus generating realistic disaster-stricken scenarios.

III. Animation of the Traffic System

The proposed system is built mainly upon two kinds of objects, the road object and the vehicle object. In order to represent the road objects, we apply a simple data structure while keeping the road curve information that is crucial to the vehicles' routes. We represent a road with an array of connected, adjacently nonparallel line segments, which are arranged to model the road's curve. Such a representation facilitates a simple principle for vehicles to move on the roads, as will be discussed in Section IV. Currently for a simpler system setup, we consider only non-crossing roads.

Each segment is represented by a segment object that stores all the information critical to the rendering of vehicle animation. The 3D coordinate system, together with a segment object, is illustrated in Figure 1. As shown in Figure 1, a segment object has the following components: (a) P_1, P_2: the coordinates of the two ends of the segment; (b) L_1: the length of the segment; (c) L_2: the length of the segment's shadow on the ground; (d) L_3: the length of the segment's shadow on the X axis; (e) θ_V: the angle between the segment and the $X-Z$ plane (vehicle's vertical rotation relative to the ground) and (f) θ_H: the angle between the segment and $X-Y$ plane (vehicle's horizontal rotation relative to $X-Y$ plane). P_1 and P_2 are read from road data files or specified manually. The equations for initializing the different components' values are presented in Equations 1 to 5.

$$L_1 = \parallel P_1 P_2 \parallel \tag{1}$$

$$L_2 = \sqrt{(P_2.x - P_1.x)^2 + (P_2.z - P_1.z)^2} \tag{2}$$

$$L_3 = P_2.x - P_1.x \tag{3}$$

$$\theta_V = \arccos(L_2/L_1) \tag{4}$$

$$\theta_H = \arccos(L_3/L_2) \tag{5}$$

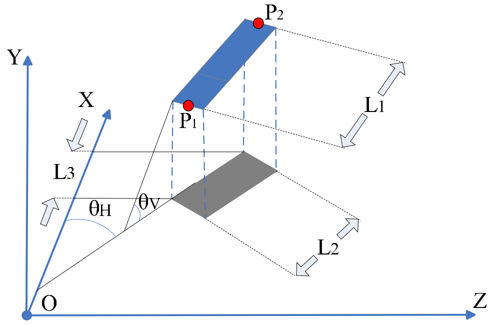


Figure 1. Segment Object Setup in 3D Coordinate System

$P_1.x$, $P_1.y$, $P_1.z$ and $P_2.x$, $P_2.y$, $P_2.z$ denote the x, y and z components of P_1 and P_2 respectively. Equation 1 calculates the distance between the two segment ends. Equation 2 calculates the shadow's length on the $X-Z$ plane. Equation 3 calculates the shadow's length on axis X. Equation 4 calculates the vertical direction angle. And Equation 5 calculates the horizontal direction angle (the angle between the segment's shadow and positive direction of x-axis).

IV. Vehicle Animation

This section discusses the object-based vehicle model and vehicle movement animation on the road structure discussed in Section III.

A. Object-based Vehicle Model

Each vehicle acts as a vehicle object in the system. There are two sets of parameters in each object. The first set has five static parameters, viz model type, scaling, default direction and normal speed, which are initiated once. The second set has five dynamic parameters, which are constantly updated during animation. They are (a) *Current_Segment_ID*: the ID of the segment the vehicle is in for the current frame; (b) *Offset_in_Segment*: the distance between the current segments starting point and the vehicle; (c) *Vertical_Direction*: the vertical angle between vehicle's current direction and default direction; (d) *Horizontal_Direction*: the horizontal angle between vehicle's current direction and default direction; and (e) *Current_Speed*: vehicle's actual speed for the current frame.

B. Principle of Vehicle Movement

During animation, each vehicle's position in the next frame is obtained by calculating its relative movement from the current frame. The relative movement is obtained by multiplying the vehicle object's *Current_Speed* with the elapsed time since the vehicle was in the last frame. The variable *Offset_in_Segment* is increased by the above calculated relative movement. If the modified offset exceeds the length (L_1) of the current segment, indicated by *Current_Segment_ID*, the system subtracts L_1 from the modified offset. This becomes the vehicle's offset in the next segment. Such deduction iterates until the remaining offset no longer exceeds the length of the segment considered. The final segment, thus reached, then becomes the vehicle's target segment. Table I presents the algorithm for finding vehicle's target segment with regard to the increased *Offset_In_Seg* (*Offset_In_Segment*).

Table I
Algorithm to Find Vehicle's Target Segment

```
void **Find_Target_Segment**(){
//find vehicle's target segment with regard to
//increased Offset_In_Seg.
    while (Offset_In_Seg > Curr_Seg_Length){
        Offset_In_Seg ←
        (Offset_In_Seg − Curr_Seg_Length);
        Curr_Seg ← Next_Seg;
    }
    Target_Seg ← Curr_Seg;
}
```

After the target segment is decided, the system needs to find out the vehicle's final position in that segment with the updated *Offset_In_Seg*. The calculation is actually done by a pair of similar triangles, $\triangle P_1P_vA$ and $\triangle P_1P_2B$ as shown in Figure 2. In Figure 2, P_v stands for the vehicle's target position in Segment S while P_1 and P_2 stand for the two ends of Segment S. Equations 6 to 8 calculate P_v's y coordinate. As a property of similar triangles, we have the proportion relationship in Equation 6, which can be extended as in Equation 7. Since $A.y = B.y = P_1.y$, we replace $A.y$ and $B.y$ with $P_1.y$ and get the expression of $P_v.y$ as presented in Equation 8. $P_v.x$ and $P_v.z$ can be obtained by a similar process.

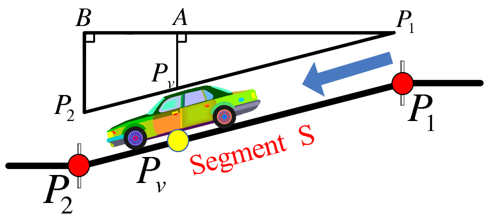


Figure 2. Deciding A Vehicle's Target Position Within A Segment

$$\frac{\parallel AP_v \parallel}{\parallel BP_2 \parallel} = \frac{\parallel P_1P_v \parallel}{\parallel P_1P_2 \parallel} \tag{6}$$

$$\frac{A.y - P_v.y}{B.y - P_2.y} = \frac{Offset\_in\_Segment}{Length\_of\_Segment} \tag{7}$$

$$P_v.y = \frac{Offset\_in\_Segment}{Length\_of\_Segment}(P_2.y - P_1.y) + P_1.y \tag{8}$$

C. Principle of Vehicle Model Rotation in Accordance with Road Direction

In the proposed traffic system, vehicle models are required not only to move in a way that matches the road's curve but also to be coherent with the direction of the current road segment. Whenever a vehicle enters a new segment, the system updates both its horizontal and vertical direction angles to be the same as the horizontal and vertical angles of the corresponding segment. Thus, the vehicle needs to be rotated. If the rotation operations on vehicle models are conducted about absolute axes, careful attention must be paid to the order in which the rotation is to be conducted. By analyzing how the two rotations affect the model, it is found that the rotation of the vertical axis should be done prior to the horizontal axis. However, if a vehicle has been rotated in the previous segment, the previous rotation should be undone before the vehicle undergoes the rotation in the current segment. In order to undo the previous rotation, the vehicle should be rotated in a reverse order, i.e. first horizontally and then vertically.

V. Additional Features

Traffic systems designed for high quality rendering situations such as storm surge impact simulations, attach great importance to interaction between objects within the system.

This section discusses the three additional features of the proposed traffic system. These features serve to render interaction within or beyond the traffic system and thus increases the creditability of the entire animation scene.

A. Interaction between Vehicles

Vehicles in a traffic system, if given different speeds to emulate real-life situations, should be kept a certain safe distance away from each other. This is done in order to ensure that they don't overlap/collide with each other. For the time being, overtaking is not allowed in the system. So we focus on the relative speed between successive vehicles. Two components, *Acceleration* and *Alert_Distance*, are added to a vehicle object. A faster car must get slowed down when it is within its *Alert_Distance* from the car in front of it. It is done by setting the car's *Acceleration* to a negative value (-1). The deceleration continues in a couple of frames until the two successive cars run at the same speed. Table II presents the algorithm for speed control between vehicles.

B. Vehicle Response in Surge Flooding

As mentioned above, one of the main contributions of this paper is the designing of a traffic system that responses to surge effects during hurricanes. The data about the height of the storm surge is obtained from NOAA and the animation dynamically reflects real-time values. Five additional parameters are added to a vehicle object to facilitate this feature. They are *Shallow_Threshold*, *Deep_Threshold*, *Paralyzing_Threshold*, *Shallow_Target_Speed* and *Deep_Target_Speed*. Vehicles will slow down and may stop when moving into flooded area. At each frame, a water height parameter received from NOAA [3] is checked and a corresponding deceleration value is determined. This continues until the target speed is reached. If the water height just passes *Shallow_Threshold*, *Acceleration* is set to -1 and the car moves at *Shallow_Target_Speed* with a wake effect rendered at its wheels. When the water height passes *Deep_Threshold*, *Acceleration* is set to a greater negative (-2). *Deep_Target_Speed* is usually set to zero, which means the vehicle will stop eventually. When the water starts to ebb, vehicles may restart if the water height hasn't passed the *Paralyzing_Threshold*. Table III presents the algorithm for vehicle response in storm surge.

C. Automatic Distribution of Lamp Posts along Roads

Any realistic simulation of a traffic system is incomplete without a lighting system or lamp posts along the road. Lamp posts can be planted at a certain position by manual specification. However, such a method becomes impractical when a large number of lamp posts are considered. Therefore, a method is designed to automatically distribute lamp posts with equal separation distance along a road. The important issue is to efficiently place lamp posts perpendicular to the current road segment with a distance from the road center. Figure 3 illustrates the solution to this issue.

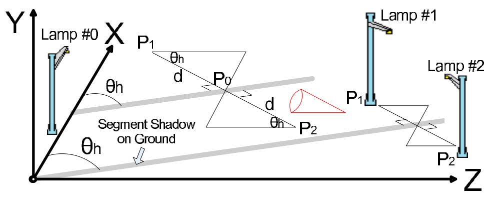


Figure 3. Plant Lamps Perpendicular to the Road

In Figure 3, the Grey stripe is a road segment's shadow on the ground. θ_h stands for the segment's horizontal direction. Lamp #0 is the default status for lamps. $\overrightarrow{P_1P_0}$, $\overrightarrow{P_2P_0}$ stand for the direction of Lamp #1 and Lamp #2 respectively. In order for the two lamps to face perpendicular to the segment's shadow, Lamp #1 is rotated by ($\theta_h + 90^o$) and Lamp #2 is rotated by ($\theta_h - 90^o$).

Table II
ALGORITHM TO INTRODUCE SPEED CONTROL BETWEEN VEHICLES

```
void **Speed_Control_Between_Vehicles**(){
//check Distance (distance from next vehicle)
//and adjust Acceleration accordingly.
    if (Distance < Alert_Distance){
        if (Current_Speed >
            next vehicle's Current_Speed)
          Acceleration ← -1;
        else
          Acceleration ← 0;
    }}
```

In order for the lamps to stay at a distance d from P_0, a line rotation is executed, which is further analyzed in Equation 9 to 12. The variables have the same meaning as in Figure 3. Equation 9 and 11 calculate the x coordinate of P_1 and P_2 while Equation 10 and 12 calculate the z coordinate of P_1 and P_2. The y coordinate for a lamp is obtained from an elevation map [13] with regard to the x and z coordinates.

$$P_1.x = P_0.x + d.sin(\theta_h) \tag{9}$$

$$P_1.z = P_0.z - d.cos(\theta_h) \tag{10}$$

$$P_2.x = P_0.x - d.sin(\theta_h) \tag{11}$$

$$P_2.z = P_0.z + d.cos(\theta_h) \tag{12}$$

VI. EXPERIMENTS

The proposed system is experimented within the 3D Visualization and Animation of Storm Impact Project [3] [4] environment, which is built upon VTP [14] platform. Experiments are conducted on 3 South Florida locations,

Table III
ALGORITHM TO INTRODUCE VEHICLE'S RESPONSE TO SURGE EFFECT

```
void **Respond_to_Flooding_Water**(){
//check Water_Height (current water height)
//and make corresponding response.
    if (Water_Height $>$ Deep_Threshold){
        if (Water_Height $>$ Paralyzing_Threshold)
            isParalyzed $\leftarrow$ true;
        if (Current_Speed $>$ Deep_Water_Speed)
            Acceleration $\leftarrow$ -2;
        elseif (Acceleration $>$ 0)
            Acceleration $\leftarrow$ 0;
    }
    elseif (Water_Height $>$ Shallow_Threshold){
        if (Current_Speed $>$ Shallow_Water_Speed)
            Acceleration $\leftarrow$ -1;
        elseif (Acceleration $>$ 0)
            Acceleration $\leftarrow$ 0;
    }}
```

namely South Beach, Key Biscayne and Ft. Lauderdale. The animation is rendered at the resolution of $1024X768$ with Intel Core 2 Quad CPU Q9550 @2.83GHz, 4092MB RAM memory and NVIDIA GeForce 9800 GT as video device (Sys I). However, we also tested the animation system on a lower configuration machine with 2 Pentium 4 CPU at 3.80Ghz, 3582MB RAM and RADEON X600 (Sys II). The terrains in the animation were constructed with LIDAR data [13] collected for corresponding locations.

Figure 4 and 5 present the animation effect of the proposed system on Miami South Beach and Key Biscayne with (a) for before-storm effect and (b) for after-storm effect. Figure 6 presents the effect of automatic distribution of lamps. The efficiency of the animation system can be evaluated by the number of reduced FPS (Frame Per Second) when the traffic system is added to the original 3D environment. For each location, we vary the number of vehicle models to be rendered and observe the reduction in frame rate. Table IV presents the frame rate for Sys I when traffic system is rendered in different locations.

Table IV
FRAME RATE (FPS) FOR SYS I

Num of Vehicles	South Beach	Key Biscayne	Ft. Lauderdale
0	7	16	8
10	7	X	X
20	7	X	X
30	7	X	8
40	6	X	8
50	X	X	8
60	X	16	8
70	X	16	8
80	X	16	8
90	X	16	X
100	X	15	X

The number of vehicles added to a specific location is chosen from a range which depends on the total road length of that location. If a cell of Table IV has an 'X', that means the number of vehicles is beyond the range specified for the particular location and no valid result can be displayed. For Sys II, the average reduction in frame rate, while the traffic system is added, is 1 FPS/40 vehicles. The frame rate given for 0 vehicles indicates the status when the traffic system is not activated. Besides, since each location has a unique setup of models and engines, the primary frame rate varies between locations. Comparing between each location, we can see that the addition of the proposed traffic system doesn't impose a distinct reduction on the frame rate. So we can conclude that the proposed traffic system doesn't introduce any considerable computational workload to the animation system.

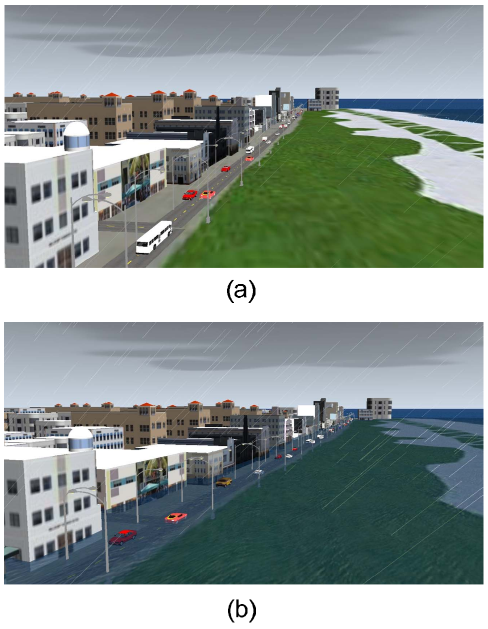
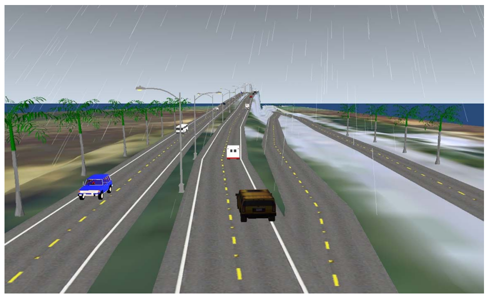


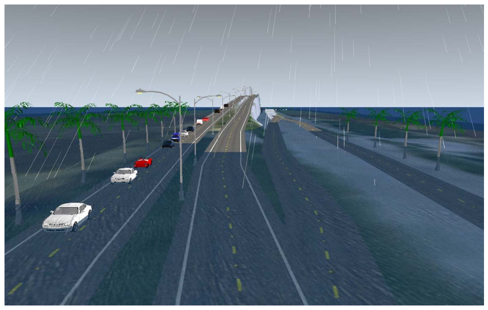
Figure 4. (a) Miami South Beach before Storm Surge (b) Miami South Beach after Storm Surge

VII. CONCLUSION

In this paper, we proposed an efficient 3D traffic animation system that works on road segmentation and supports vehicle interaction and surge flooding response. The segment-based decomposition leads to an easy representation of roads. Vehicle movement is animated by finding the target segment and in-segment offset at each frame and rotation is done according to segment direction. Speed control is conducted by inspecting the vehicles speed, acceleration and distance from the next vehicle. Storm surge response is realized with inputs from meteorological source. The system also supports automatic lamp distribution along roads. The experimental results demonstrate encouraging outcome that the system runs at a low computation overhead and renders realistic animation effect of traffic in flooded situations.



(a)



(b)

Figure 5. (a) Key Biscayne before Storm Surge (b) Key Biscayne after Storm Surge

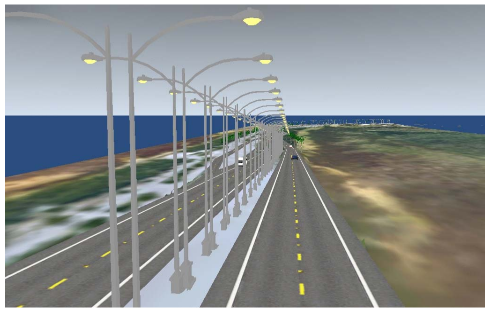


Figure 6. Street Lamp Distribution

As part of our future work, we plan to expand the traffic system to be graph based such that complicated road network could be more easily handled.

Acknowledgment

This research project was supported in part by a grant from the National Oceanic and Atmospheric Administration (NOAA). The authors would also like to thank Junior Betancourt and Jairo Pava for assisting in the development process.

References

[1] B. Delaney, "Visualization in urban planning: they didn't build LA in a day," *IEEE Computer Graphics and Applications*, vol. 20, no. 3, pp. 10–16, 2000.

[2] A. Behzadan and V. Kamat, "Simulation and visualization of traffic operations in augmented reality for improved planning and design of road construction projects," in *Simulation Conference*, 2008, pp. 2447–2454.

[3] S.-C. Chen, K. Zhang, and M. Chen, "A real-time 3d animation environment for storm surge," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2003, pp. 705–708.

[4] K. Zhang, S.-C. Chen, P. Singh, K. Saleem, and N. Zhao, "A 3d visualization system for hurricane storm-surge flooding," *IEEE Computer Graphics and Applications*, vol. 26, no. 1, pp. 18–25, 2006.

[5] M. Turk, D. Morgenthaler, K. Gremban, and M. Marra, "Vits-a vision system for autonomous land vehicle navigation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 342–361, 1988.

[6] A. Waxman, J. Lemoigne, L. Davis, B. Srinivasan, T. Kushner, L. Eli, and T. Siddalingaiah, "A visual navigation system for autonomous land vehicles," *IEEE Journal of Robotics and Automation*, vol. 3, no. 2, pp. 124–141, 1987.

[7] Y. Yagi, M. Brady, Y. Kawasaki, and M. Yachida, "Active contour road model for smart vehicle," in *Proceedings of 15th International Conference on Pattern Recognition*, 2000, pp. 811–814.

[8] E. Dickmanns and B. Mysliwetz, "Recursive 3-d road and relative ego-state recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 199–213, 1992.

[9] K. Kluge and C. Thorpe, "Representation and recovery of road geometry in YARF," in *Proceedings of the Intelligent Vehicles*, 1992, pp. 114–119.

[10] T. Wang, S. Tang, and P. Pang, "3d urban traffic system simulation based on geo data," in *2nd International Conference on Information Technology: Research and Educations*, 2004, pp. 59–63.

[11] P. Ioannou, Z. Xu, S. Eckert, D. Clemons, and T. Sieja, "Intelligent cruise control: theory and experiment," in *Proceedings of the 32nd IEEE Conference on Decision and Control*, 1993, pp. 1885–1890.

[12] Y. Kuroda, T. Ura, and K. Aramaki, "Vehicle control architecture for operating multiple vehicles," in *Proceedings of the 1994 Symposium on Autonomous Underwater Vehicle Technology*, 1994, pp. 323–329.

[13] K. Zhang, S.-C. Chen, D. Whitman, M. Shyu, J. Yan, and C. Zhang, "A progressive morphological filter for removing non-ground measurements from airborne LIDAR data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 4, pp. 872–882, 2003.

[14] "Virtual terrain project," http://vterrain.org/, 2009.