Intelligent Computational Transportation

Yuming Zhang

Intelligent Computational Transportation

by

Yuming Zhang

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Computer Engineering

The University of New Mexico

Albuquerque, New Mexico

July, 2018

Acknowledgments

I give my deepest love and unbounded gratitude to my family, Yunyun, Sherry, Leo, Albert, and my parents. They are my motivation to keep going forward firmly and fearlessly. I would like to thank my coworkers, Steven Garcia, Ran Luo, Qiong Wu, Cong Chen, for their open discussion and help on the projects. I would also like to thank the committees, Dr. Rafael Fierro, Dr. Marios Pattichis, Dr. Wei Shu, and Dr. Guohui Zhang, for their kindly suggestions and professional comments. Many thanks are also given to my advisor, Dr. Yin Yang. This dissertation would not happen without his help and support.

Intelligent Computational Transportation

by

Yuming Zhang

Ph.D., Computer Engineering, University of New Mexico, 2018

Abstract

Transportation is commonplace around our world. Numerous researchers dedicate great efforts to vast transportation research topics. The purpose of this dissertation is to investigate and address a couple of transportation problems with respect to geographic discretization, pavement surface automatic examination, and traffic flow simulation, using advanced computational technologies.

Many applications require a discretized 2D geographic map such that local information can be accessed efficiently. For example, map matching, which aligns a sequence of observed positions to a real-world road network, needs to find all the nearby road segments to the individual positions. To this end, the map is discretized by cells and each cell retains a list of road segments coincident with this cell. An efficient method is proposed to form such lists for the cells without costly overlapping tests. Furthermore, the method can be easily extended to 3D scenarios for fast triangle mesh voxelization.

Pavement surface distress conditions are critical inputs for quantifying roadway infrastructure serviceability. Existing computer-aided automatic examination techniques are mainly based on 2D image analysis or 3D georeferenced data set. The disadvantage of information losses or extremely high costs impedes their effectiveness and applicability. In this study, a cost-effective Kinect-based approach is proposed for 3D pavement surface reconstruction and cracking recognition. Various cracking measurements such as alligator cracking, traverse cracking, longitudinal cracking, etc., are identified and recognized for their severity examinations based on associated geometrical features.

Smart transportation is one of the core components in modern urbanization processes. Under this context, the Connected Autonomous Vehicle (CAV) system presents a promising solution towards the enhanced traffic safety and mobility through state-of-the-art wireless communications and autonomous driving techniques. Due to the different nature between the CAVs and the conventional Human-Driven-Vehicles (HDVs), it is believed that CAV-enabled transportation systems will revolutionize the existing understanding of network-wide traffic operations and re-establish traffic flow theory. This study presents a new continuum dynamics model for the future CAV-enabled traffic system, realized by encapsulating mutually-coupled vehicle interactions using virtual internal and external forces. A Smoothed Particle Hydrodynamics (SPH)-based numerical simulation and an interactive traffic visualization framework are also developed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The development of modern transportation takes great advantage of emerging computational technologies. A huge amount of traffic data can be collected from various advanced sensing technologies, inductance loop detectors, surveillance cameras, Global Positioning System (GPS) modules, etc. The data provides detailed instant traffic states and facilitates traffic control and prediction. In the infrastructure maintenance area, autonomous road analyzers, such as a vehicle equipped with a laser-based scanning system, are widely used to help transportation agencies monitor and evaluate the road surface distress conditions and determine the infrastructure serviceability. Recent Connected Autonomous Vehicle (CAV) technologies evolve very quickly and present an attractive solution to augmented traffic safety and mobility. On the other hand, the deployment of advanced computational technologies brings new challenges to transportation. To fully dig out the huge information in the big traffic data, people might have to dedicate great effort and develop new analysis methods. Current autonomous road analyzers are highly expensive and consume a considerable part of the annual transportation budget. As the urban road network keeps expanding, low-cost road analyzers need to be developed. Equipped with advanced communication technologies, CAVs are mutually-coupled which makes them

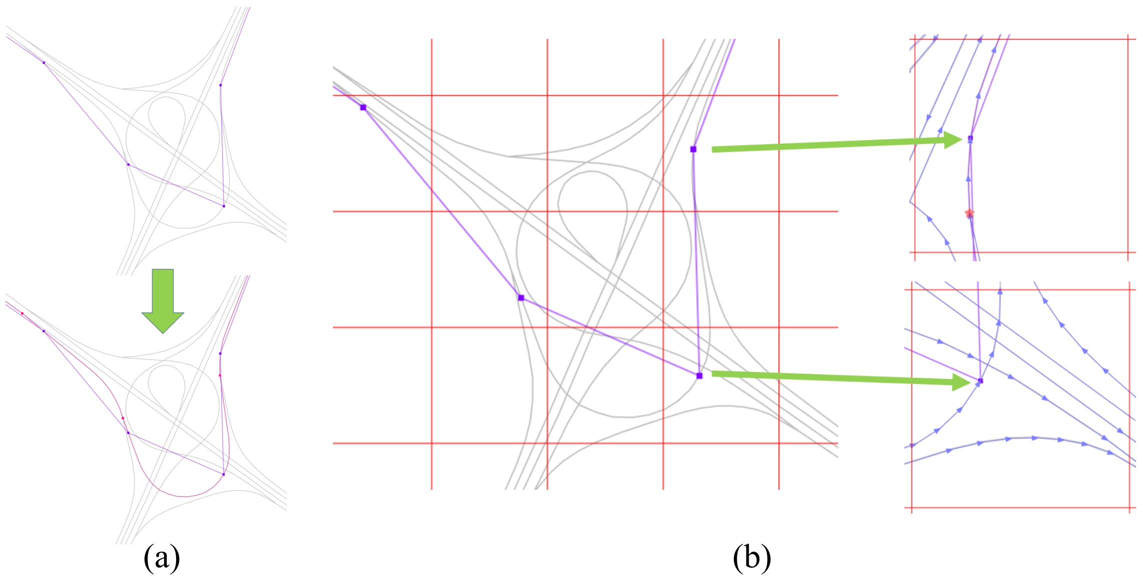


Figure 1.1: The map matching algorithm (a) needs to efficiently find all the nearby road segments to the given positions (b).

different from conventional Human Driven Vehicles (HDVs). Consequently, current traffic flow models might not be suitable for simulating CAV-enable traffic systems. New traffic flow theory for CAV-enable traffic needs to be established. The purpose of this dissertation is to investigate and address a couple of transportation problems using advanced computational technologies.

Discretization is essential to computer-related technologies, because computers deal with digital quantities. For instance, a real-world curved road is represented by a sequence of line segments, or road segments, in Geographic Information System (GIS) data. Many applications in transportation need a discretized map such that local geographic information can be accessed efficiently. Map matching [LZZ$^+$09], an algorithm that aligns a sequence of position, or a trajectory, to a road network, is one of those application (Fig. 1.1 (a)). One key step of the algorithm is to efficiently find all the nearby road segments to a given position. The solution is easy since one can discretize the map by cells and let each cell retain a list that contains all road segments coincident with this cell (Fig. 1.1 (b)). Therefore, once a position is given, the cell is determined immediately and so as the list. To generate the list, it is conventional to apply overlapping tests between the cells and the road segments. In Chapter 2, we propose an efficient discretization method avoiding the costly overlapping tests. Furthermore, the method can be extended to 3D scenarios for the surface voxelization of geometrically complex models [ZGX+17]. Unlike recent techniques relying on triangle-voxel intersection tests, our algorithm exploits the conventional parallel-scanline strategy. Observing that there does not exist an optimal scanline interval in general 3D cases if one wants to use parallel voxelized scanlines to cover the interior of a triangle, we subdivide a triangle into multiple axis-aligned slices and carry out the scanning within each polygonal slice. The theoretical optimal scanline interval can be obtained to maximize the efficiency of the algorithm without missing any voxels on the triangle. Once the collection of scanlines are determined and voxelized, we obtain the surface voxelization. We fine tune the algorithm so that it only involves a few operations of integer additions and comparisons for each voxel generated. Finally, we comprehensively compare our method with the state-of-the-art method in terms of theoretical complexity, runtime performance and the quality of the voxelization on both CPU and GPU of a regular desktop PC, as well as on a mobile device. The results show that our method outperforms the existing method, especially when the resolution of the voxelization is high.

Pavement surface distress conditions are critical inputs for quantifying roadway infrastructure serviceability. Numerous computer-aided automatic examination techniques have been deployed for pavement distress condition assessments, such as digital image processing methods. However, their effectiveness and applicability are impeded due to information losses in 2D image combination processes or extremely high costs in 3D geo-referenced data set. In Chapter 3, a cost-effective Kinect-based approach is presented for 3D pavement surface reconstruction and cracking recognition. We propose a comprehensive computational solution for the detection and recognition of pavement distress feature identification [ZCW+18]. Various cracking measurements such as alligator cracking, traverse cracking, longitudinal cracking, etc. are identified and recognized for their severity examinations based on associated geometrical features. The experimental results indicate that this method is effective in reducing data collection costs and extracting analytical information on pavement cracking measurements. The research findings confirm that the proposed approach provides a viable, applicable solution to an automatic pavement surface condition detection and evaluation. The proposed methodology is transferable for pavement surface reconstruction and distress condition detection based on the other 3D cloud point data. It provide an alternative inexpensive complement to existing pavement examination methodologies.

Recent technology advances significantly push forward the development and the deployment of the concept of *smart*, such as smart community and smart city. Smart transportation is one of the core components in modern urbanization processes. Under this context, the CAV system presents a promising solution towards the enhanced traffic safety and mobility through state-of-the-art wireless communications and autonomous driving techniques. Being capable of collecting and transmitting real-time vehicle-specific, location-specific, and area-wide traffic information, it is believed that CAV-enabled transportation systems will revolutionize the existing understanding of network-wide traffic operations and re-establish traffic flow theory. In Chapter 4, we develops a new continuum dynamics model for the future CAV-enabled traffic system, realized by encapsulating mutually-coupled vehicle interactions using virtual internal and external forces [ZZFY18]. Leveraging the Newton's second law of motion, our model naturally preserves the traffic volume and automatically handles both the longitudinal and lateral traffic operations due to its two-dimensional nature, which sets us apart from the existing macroscopic traffic flow models. Our model can also be rolled back to handle the conventional traffic of human drivers, and the experiment shows that the model describes the real-world traffic behavior well. Therefore, we consider the proposed model a complement and generalization of the existing traffic theory. We also develop a Smoothed Particle Hydrodynamics (SPH)-based numerical simulation and an interactive traffic visualization framework. By posing user-specified external constraints, our system allows users to visually understand the impact of different traffic operations interactively.

Chapter 2

Efficient Discretization

2.1 Introduction

Real world geometries have a diverse range of forms and shapes, usually consisting of various kinds of primitives like lines, triangles, polygons, curved surfaces, etc. In order to visualize, animate, render and analyze such geometries with digital computers, a discrete representation is essential. Voxelization, as one of the most widely used discretizing approaches, converts a continuous geometry into a set of volumetric pixels or *voxels* which best approximates the original shape. Voxelization plays a fundamental role in computer graphics, and it often stands as an important geometric pre-processing step in many applications, such as virtual reality [KJ+01], medical imaging/visualization [KK99], global rendering [LHLW10], collision detection [Ber97, HK97, NSSL13], computer animation or simulation [ZCK98, JBT04, LFWK05, ZD17, DKB+16], and other interesting areas [ZSMS14]. A *surface voxelization* of a 3D model produces a set of boxes/voxels that encapsulates its geometric boundary, which is often represented as a triangle mesh in computer graphics. We evaluate voxelization algorithms by their *efficiency*, *accuracy*, *separability* and

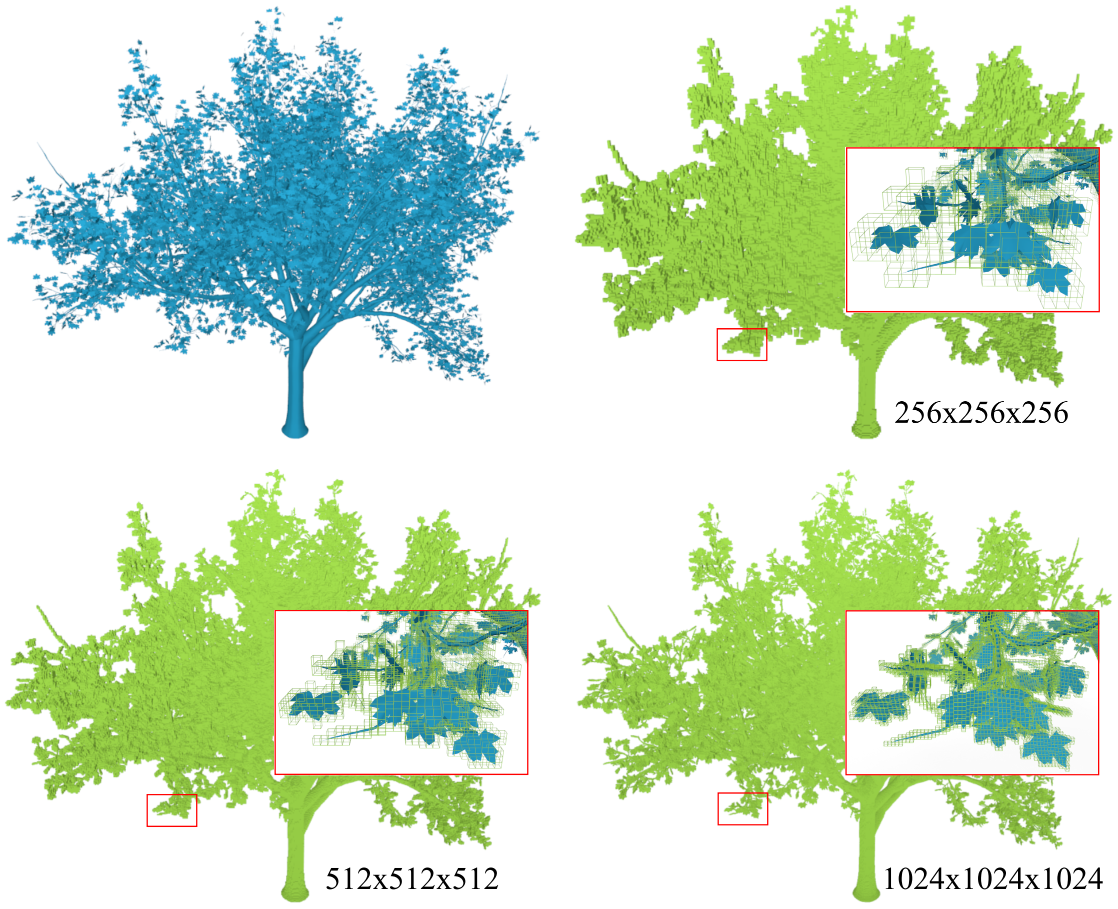


Figure 2.1: The results of the voxelization of a tree model (842K triangles) using the proposed method under the resolutions of 256, 512 and 1024 respectively.

minimality, following the framework of Cohen-Or and Kaufman [COK95]. Many existing voxelization algorithms [SS10] [Pan11] are based on *overlap tests*, wherein every potential voxel candidate undergoes a sequence of tests to determine whether it intersects a triangle. These methods produce *super covers* of input models (see Sec. 2.3 for a quick terminology review). A drawback of the methods is that the triangle-voxel intersection test could be relatively expensive, compared to the *scan-conversion* algorithms [Kau87,Kau88,KS87]. The 3D scan-conversion algorithms are extensions of their 2D counterparts (i.e. the famous Bresenham's algorithm [Bre65] and very efficient. However, due to the complication of 3D scenarios, the resulting voxelization produced by these algorithms is only a subset of the cover of the original model.

In this chapter, we propose a new algorithm that can produce a cover using parallel scanlines. A cover is *N-tunnel-free*, which is an important feature of a high-quality surface voxelization. The biggest challenge in our method is to determine an appropriate set of parallel scanlines. The scanlines should not miss any voxel on the cover, and the distance between two consecutive scanlines or the *scanline interval* (SI) should be optimal in order to achieve high efficiency. We will show that there does not exist a “gold standard” allowing us to set a constant SI in a general 3D case. Instead, our method subdivides an input triangle into multiple axis-aligned slices. The voxelization of each slice degenerates to a 2D case. We derive a theoretically optimal SI for setting up the scanlines for each slice (see detailed explanation in Sec. 2.4.2). We further derive an integer-only version of the algorithm with minor accuracy compromise, and an enhanced version for generating the super cover. We test our algorithm on various 3D models with both CPU and GPU with a desktop computer, as well as the mobile platform of IOS device (an Apple iPhone 6). Experiments, for example, voxelization of a tree model (Fig. 2.1), show that the proposed method presents a good performance, especially for a high-resolution voxelization. To the best of our knowledge, our method is the first one to generate a N-tunnel-free cover or the super cover using the scanline strategy.

2.2 Related Work

The *separating axis theorem* (SAT) [GLM96, AMHH08] provides a general guideline for an overlap test between two convex polygons: the triangle-box overlap test is simply a base case in the SAT [Ebe01]. Akenine-Möller [AM05] adopted a standard triangle-box overlap test following the SAT, which consists of 13 sub-tests: three for the box against the minimal box of the triangle, one for overlap test between the box and the plane determined by the triangle, and nine for the projections of the triangle against the box. The triangle intersects the box if it passes all the tests. In a recent contribution, Schwarz and Seidel [SS10] provided the *sufficient and necessary condition* of the box-triangle intersection tests: 1) the box intersects the triangle's plane; and 2) the projections of the box and the triangle overlap on all of the three coordinate planes. They use nine edge functions to evaluate the second condition, which essentially correspond to the nine sub-tests in [AM05]. They also improved the algorithm by reducing the number of candidate voxels and skipping unnecessary tests based on the observation that a triangle is of at most three-voxel thickness in its *dominant axis* direction of the triangle's normal. Based on their method, Pantaleoni [Pan11] further reduced the number of candidate voxels in the inner loop of the 2D projection overlap test by computing a tighter bound. Crassin and Green [CG12] presented a simple voxelization pipeline basically following the work in [SS10, Pan11]. The resulting voxelization is not a correct 6-tunnel-free one, because only the coverage of the center of each voxel is tested against the triangles to generate fragments. They employed the idea in [HAMO05] to fix the problem. However, the method can not generate super covers because the voxels captured at the triangle edges could be redundant.

Overlap tests form the basis of many existing rasterization algorithm. McCool and colleagues [MWM01] examined four corners of a pixel tile against each edge of a triangle by evaluating the signs of its three edge functions: a point is inside a triangle if and only if all of the three edge functions at the point are positive. A conservative 2D rasterization algorithm was presented in [Pin88, AMA05], which modified the triangle setup and selected a different evaluation point to reduce the computation. Haines and Wallace [HW94] observed that the overlap test between a box and a plane can be done by projecting the box to the diagonal that best aligns with the plane normal, and only two corresponding corners would be involved in the test. It is suggested that if the entire box is in either the positive or the negative half-plane, only one of the box corners needs to be tested.

Instead of using overlap tests, Huang and colleagues [HYFK98] voxelized a surface by evaluating a distance threshold. If the distance from voxel center to the triangle is smaller than a certain threshold value, an overlap is determined. Varadhan and colleagues [VKK$^+$03] presented an algorithm computing the *max-norm* distance between voxels and other geometric primitives, which was formulated as an optimization problem. Brimkov and Barneva [BB02] presented a nice surface voxelization namely the *graceful plane*, which is 6-tunnel-free and jump free. Graceful planes are the thinnest possible discrete voxelizations in which any geometry primitives are connected sets of voxels. Fei and colleagues [FWC12] proposed a point-tessellated voxelization method. Taking advantage of the powerful tessellator in GPU hardware, the method efficiently generates watertight voxelizations. The resulting voxelizations approximate the ground truth (i.e. super covers) well and are suitable for applications where accuracy is not the major concern, such as video games and virtual realities. Chao and colleagues [CWDY15] set up a set of sampling points of the input triangle. The voxelization of the triangle is simply the union of all the point voxelization. The method also produces an approximation and does not guarantee to generate a cover.

The 3D/2D line voxelization algorithms lie in the most inner loop of our algorithm and must be efficent. Liu and Chen [LC02] extended 2D Bresenham's algorithm for 3D line segments. Au and Woo [AW11] investigated the 3D Bresenham's algorithm using the Voronoi diagram. Our method borrows the ideas in these 3D line rasterization techniques, which generate the line voxelization by incrementally evaluating stepping parameters along the straight line segment to determine the corresponding voxel sequence [AW$^+$87, COK97, HCTS10]. We improve this algorithm so that only integer operations are involved, which could further speed up the computation with the support of dedicated hardware [SBV91, LM00].

As a fundamental algorithm, voxelization has been extensively implemented and tested on modern GPUs architectures. Dong and colleagues [DCB+04] proposed a fast voxelization algorithm on the GPU for complex polygon models, which achieved a real-time frame rate. GPU-accelerated algorithm [ED08, IDC09] as well as GPU-oriented data structures [ZHWG08, ZGHG11] have been researched in order to optimally utilize the hardware resources. Our method is parallelizable and has been implemented using `nVidia CUDA` in our experiments.

2.3 Terminology Review

We briefly review some useful terminologies [COK95, SS10, COK97, ZCEP07] regarding the properties and quality of a voxelization. The generated voxel is assumed to have a unit size in all of its x, y and z directions, and we use $\mathbb{Z}^3$ to denote the set consisting of all the integer coordinates or *grid points* corresponding to the centers of all the voxels. Hereinafter, we use bold lowercase letters, e.g. $\mathbf{p}(p_x, p_y, p_z)$ to denote a point defined in $\mathbb{R}^3$ with real coordinates p_x, p_y and p_z, and bold uppercase letters like $\mathbf{P}(P_x, P_y, P_z)$ to denote a grid point or a voxel with integer indices of P_x, P_y and P_z in $\mathbb{Z}^3$.

Two neighboring voxels are *26-adjacent* if they share a face, an edge or a corner. Similarly, two voxels are *18-adjacent* if they are connected by a face or an edge, or *6-adjacent* if connected by just a face. An *N-path* is a voxel sequence in which any two consecutive voxels are N-adjacent, for $N \in \{6, 18, 26\}$. By connecting the centers of every two adjacent voxels along an N-path, we obtain its *polygon arc*. Let $\mathcal{S}$ be a continuous surface patch and $\mathcal{V}$ be its voxelization. We say that an N-path penetrates $\mathcal{V}$ if its polygon arc passes through $\mathcal{S}$. If there does not exist an N-path in $\mathbb{Z}^3 \setminus \mathcal{V}$ penetrating $\mathcal{V}$, $\mathcal{V}$ is called *N-tunnel-free*. If $\mathcal{S}$ is completely encapsulated by $\mathcal{V}$, and every voxel in $\mathcal{V}$ is intersecting $\mathcal{S}$, $\mathcal{V}$ is called a *cover* of $\mathcal{S}$. Intuitively, a cover with N-tunnel-free property is a *good* voxelization of the surface. [1]

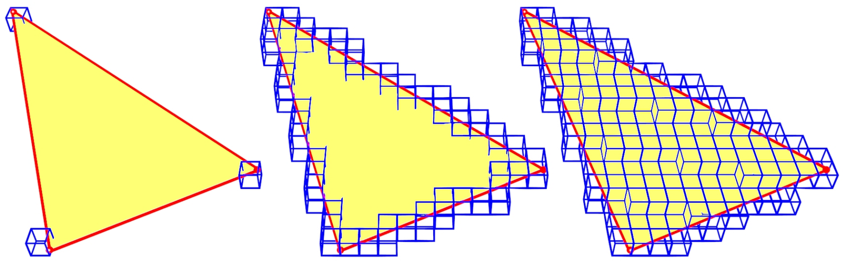


Figure 2.2: The voxelization of vertices, edges and the interior area of a triangle.

2.4 Scanline-based Voxelization

As shown in Fig. 2.2, our algorithm consists of three steps, namely the voxelization of triangles' vertices, edges, and the interior. Our voxelization is 26-tunnel-free, and it can be downgraded to 18- or 6-tunnel-free to generate thinner voxelizations if necessary. The first step of vertex or point voxelization is trivial. Given a point $\mathbf{p}_0(x_0, y_0, z_0)$, its corresponding voxel indices can be easily obtained as $\mathbf{P}_0(\lfloor x_0 + 1/2\rfloor, \lfloor y_0 + 1/2\rfloor, \lfloor z_0 + 1/2\rfloor)$. Next, we will detail the second and the third step of how to voxelize the edges and the interior of an input triangle.

[1]Strictly speaking, there are small differences between concepts of *cover*, *super cover* and *minimum cover* as discussed in existing literature [COK95]. Think of a simple 3D point coinciding with a voxel's corner. A super cover will be all the eight voxels incident to the point. Any one of these eight voxels is a minimum cover, and any subset of these eight voxels is a cover. For a closed triangularized manifold, however it is not practically necessary to differentiate these minor differences.

2.4.1 Line Voxelization

Our line voxelization is based upon the work of Amanatides and Woo [AW+87], which efficiently generates a 6-path line voxelization. We further generalize this method to an integer-only version so that only integer operations are needed.

RLV: regular line voxelization Assume that two ends of the line segment, given by $\mathbf{p}_0(x_0, y_0, z_0)$ and $\mathbf{p}_1(x_1, y_1, z_1)$, are contained by the voxels $\mathbf{P}_0(X_0, Y_0, Z_0)$ and $\mathbf{P}_1(X_1, Y_1, Z_1)$ respectively. Let $\mathbf{v} = [v_x, v_y, v_z]^\top$ be a unit vector directing from $\mathbf{p}_0$ to $\mathbf{p}_1$. If we cast a ray from $\mathbf{p}_0$ to $\mathbf{p}_1$, this ray will first intersect a facet of $\mathbf{P}_0$, and the voxel adjacent to this intersected facet will be marked as part of the final voxelization. As the ray moves forward, it will intersect $|X_1 - X_0|$ yz facets, $|Y_1 - Y_0|$ xz facets, and $|Z_1 - Z_0|$ xy facets before it reaches $\mathbf{p}_1$. We define the $x-$direction *distance function* $l^x(i)$ as the distance along the ray from $\mathbf{p}_0$ to the i^{th} yz facet, where $0 \le i \le |X_1 - X_0|$. The projection of $l^x(i)$ on the x axis is $l^x(i) \cdot |v_x|$. Recalling that the dimension of a voxel h equals to 1, we have $l^x(i) \cdot |v_x| = d_0^x + i$ or:

$$l^x(i) = \frac{d_0^x}{|v_x|} + \frac{i}{|v_x|}, \tag{2.1}$$

where d_0^x is the distance (along x axis) between $\mathbf{p}_0$ and its first intersecting yz facet, which can be evaluated as:

$$d_0^x = \begin{cases} X_0 - x_0 + \frac{1}{2} & \text{if} \quad v_x > 0 \\ \frac{1}{2} - (X_0 - x_0) & \text{if} \quad v_x < 0 \end{cases} . \tag{2.2}$$

If $v_x = 0$, we have $l^x(i) \to \infty$ which means the ray will never hit a yz facet and the line voxelization degenerates to 2D.

In each iteration, we track the *step distances*, denoted by l^x, l^y, and l^z, from the current voxel to the next yz, xz, and xy facet. Let $m^x = 1/|v_x|$, $m^y = 1/|v_y|$, and $m^z = 1/|v_z|$ denote the slopes of three distance functions. Assume that at certain step j, $l^x = l_{min} \triangleq \min\{l^x, l^y, l^z\}$, indicating that the ray will intersect a yz facet and $\mathbf{P}'(X_j + \Delta X, Y_j, Z_j)$ will be marked next. Three step distance variables for the next step can be updated incrementally as: $l^x \leftarrow l^x - l_{min} + m^x = m^x$, $l^y \leftarrow l^y - l_{min}$, and $l^z \leftarrow l^z - l_{min}$. This process stops when the ray reaches $\mathbf{p}_1$, where the last voxel generated will be $\mathbf{P}_1(X_1, Y_1, Z_1)$.

The minimum step distances are not unique if the line segment intersects a voxel at one of its edges or corners, which is referred to as a *singular point*. We can either pick an arbitrary voxel candidate with minimum step distance or pick all the voxels incident to the singular point. In the latter case, the corresponding line voxelization forms a super cover and we call this modified method *super-cover line voxelization* (SLV). Fig. 2.3 (a) and (b) show the results of RLV and SLV.

ILV: integer-only line voxelization The integer-only line voxelization eliminates runtime floating-point arithmetics in RLV. To do so, we use the grid points of $\mathbf{P}_0$ and $\mathbf{P}_1$ to approximate $\mathbf{p}_0$ and $\mathbf{p}_1$, and d_0^x, d_0^y, d_0^z in Eq. (2.2) equal to 1/2. Let $\mathbf{v}$ be $[X_1 - X_0, Y_1 - Y_0, Z_1 - Z_0]^\top$, and the distance functions are re-written as:

$$\begin{cases} l^x(i) = \dfrac{1}{|X_1 - X_0|} i + \dfrac{1}{2|X_1 - X_0|} \\ l^y(i) = \dfrac{1}{|Y_1 - Y_0|} i + \dfrac{1}{2|Y_1 - Y_0|} \\ l^z(i) = \dfrac{1}{|Z_1 - Z_0|} i + \dfrac{1}{2|Z_1 - Z_0|} \end{cases} . \tag{2.3}$$

It is noteworthy that true values of distance functions are of less interest, as we only need the relative order among l^x, l^y, and l^z to determine the next voxel. Therefore, we multiply both sides of Eq. (2.3) by a scaling factor $s = 2|X_1 - X_0||Y_1 - Y_0||Z_1 - Z_0|$ resulting in three integer-valued distance functions denoted by $L^x(i) = s \cdot l^x(i)$, $L^y(i) = s \cdot l^y(i)$, and $L^z(i) = s \cdot l^z(i)$ such that:

$$\begin{cases} L^x(i) = 2M^x i + M^x \\ L^y(i) = 2M^y i + M^y \\ L^z(i) = 2M^z i + M^z \end{cases} , \tag{2.4}$$

where $M^x = |Y_1 - Y_0||Z_1 - Z_0|$, $M^y = |X_1 - X_0||Z_1 - Z_0|$ and $M^z = |X_1 - X_0||Y_1 - Y_0|$ are all integers. Then, we can utilize three integers $L^x = s \cdot l^x$, $L^y = s \cdot l^y$ and $L^z = s \cdot l^z$ as the integer-counterparts of l^x, l^y and l^z to determine the voxels to be generated along the ray. ILV is an approximation of RLV as it forces the ends of a line segments to be at the grid points.

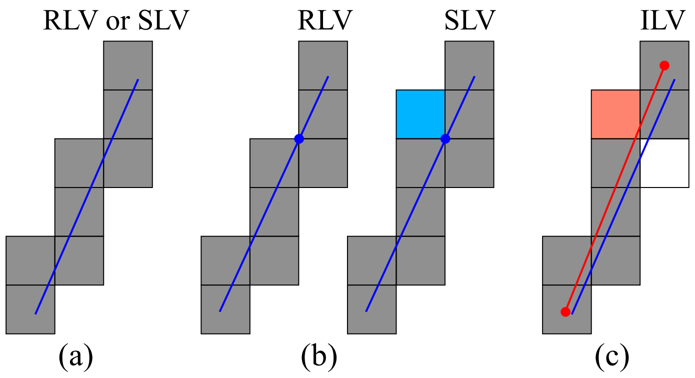


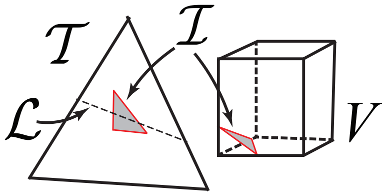
Figure 2.3: (a) RLV and SLV generate identical voxelization for a general line segment without singular points. (b) SLV captures more voxles than RLV to form a super cover if the line segment contains a singular point. (c) Small variations of the voxels generated by ILV and RLV due to the voxel center approximation.

2.4.2 Triangle Voxelization

Triangle voxelization targets the interior of an input triangle $\mathcal{T}$, and outputs a 26-tunnel-free set of voxels $\mathcal{V}$ that completely encapsulates $\mathcal{T}$. Unlike SAT-based methods, we do not perform excessive overlap tests for voxels residing in the bounding box of $\mathcal{T}$. Instead, we carefully form a set of scanline segments: the voxelization of each scanline can be obtained with RLV/ILV, and the superset of all the scanline voxelizations will be the final output of this procedure. While the intuition could lead one to select a set of parallel scanline segments to cover the triangle, it turns out that there does not exist an optimal scanline interval (SI) for the general 3D case that guarantees not missing any voxels on the cover. Thus, the performance of a 3D scanline method is often unsatisfying and slower than SAT-based methods.

In this section, we show that such technical challenge can be resolved by projecting $\mathcal{T}$ onto axis-aligned slices. We show that an optimal scanline interval is available within a 2D slice, which guides us to set the most aggressive scan strategy slice by slice. Finally, we give an integer version of this method, with minor compromises of the scanning optimality using integer-based scanline intervals.

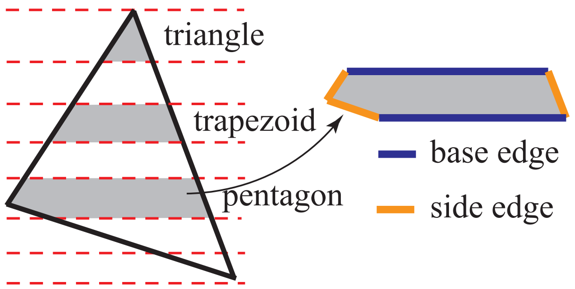
Scanline interval in 3D Let V be a voxel intersecting $\mathcal{T}$ as shown on right. We use $\mathcal{I}$ to denote the intersecting region on $\mathcal{T}$ such that $\mathcal{I} = \mathcal{T} \bigcap V$. Clearly, a 26-tunnel-free voxelization of $\mathcal{T}$ must include V. We can also learn from the line voxelization procedure that the voxelization of a scanline $\mathcal{L}$ includes V only if $\mathcal{L} \bigcap V \neq \varnothing$. As the scanline is also on the triangle i.e. $\mathcal{L} \subset \mathcal{T}$, $\mathcal{L} \bigcap \mathcal{I}$ is also non-empty:



$$\left.\begin{array}{l} \mathcal{T} \bigcap V = \mathcal{I} \\ \mathcal{L} \bigcap V \neq \varnothing \\ \mathcal{L} \subset \mathcal{T} \end{array}\right\} \Rightarrow \mathcal{L} \bigcap \mathcal{I} \neq \varnothing.$$

In other words, the scanline $\mathcal{L}$ must intersect $\mathcal{I}$ as well, in order to produce voxel V. As the intersecting region $\mathcal{I}$ could approach to an infinitesimally small area, any finite scanline interval will miss $\mathcal{I}$ in the resulting voxelization. Therefore, we say there does NOT exist an optimal SI in 3D, and one has to resort to the 2D projection to solve this problem.

Triangle splicing We first determine $\mathcal{T}$'s *dominant direction* – the coordinate axis that best aligns with the triangle's normal. Without loss of generality, assume that z axis is its dominant direction, and we re-order $\mathcal{T}$'s three vertices $\mathbf{p}_0(x_0, y_0, z_0)$, $\mathbf{p}_1(x_1, y_1, z_1)$ and $\mathbf{p}_2(x_2, y_2, z_2)$ such that $z_0 \leq z_1 \leq z_2$. Their voxelizations are denoted with $\mathbf{P}_0(X_0, Y_0, Z_0)$, $\mathbf{P}_1(X_1, Y_1, Z_1)$ and $\mathbf{P}_2(X_2, Y_2, Z_2)$ respectively. Afterwards, $\mathcal{T}$ is sliced into a set of polygons $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_{Z_2-Z_0+1}\}$ along the z axis by a series of xy planes $\{\mathcal{S}_0, \mathcal{S}_1, ..., \mathcal{S}_{Z_2-Z_0+1}\}$. $\mathcal{T}_i$ must be convex, and it could be either a trapezoid, a triangle or a pentagon as shown here. Regardless of its geometric variations, we can always group $\mathcal{T}_i$'s edges into *side edges* and *base edges*. The side edges are the ones coincide with the original edges of $\mathcal{T}$, while the base edges, are the intersections between $\mathcal{T}_i$ and $\mathcal{S}_i$. The Z index of the i^{th} plane $\mathcal{S}_i$ is $Z_0 + i + 1/2$. It is clear that each sliced polygon $\mathcal{T}_i$ is restricted to within one-voxel-thickness along the z axis. Therefore, the voxelization of $\mathcal{T}_i$ degenerates to a 2D case with a fixed Z index.



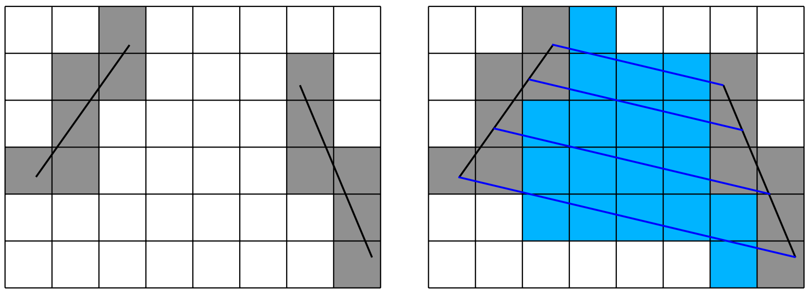


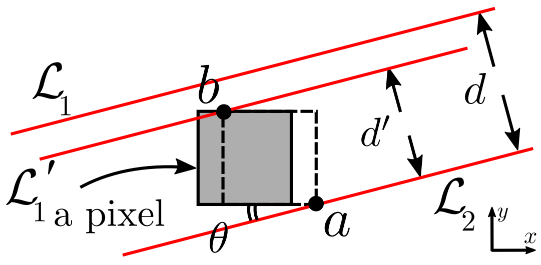
Figure 2.4: A set of parallel scanline is formed to cover the interior of $\mathcal{T}_i$. The scanline interval is set to be $\widehat{d}$.

Optimized parallel scanline For a given $\mathcal{T}_i$, our scan starts with one of its base edges as shown in Fig. 2.4. As discussed, we seek an as-sparse-as-possible scan if the resulting voxelization remains 26-tunnel-free. Hereby, we provide an upper bound of the distance between two consecutive scanlines in 2D.

Theorem 2.4.1. *In 2D, there does not exist a voxel between a pair of parallel lines that does not intersect either of them if the distance d between the lines satisfies the* *following scanning condition:*

$$d \leq \widehat{d}, \qquad \widehat{d} = h \cdot (\sin\theta + \cos\theta), \tag{2.5}$$

where h is the dimension of the voxel ($h = 1$ in our case) and θ is the smallest angle between the lines and the voxel's edges.



Proof Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be the parallel lines and assume their distance satisfies Eq. (2.5): $d \leq \widehat{d}$. Assume that there exists a voxel between $\mathcal{L}_1$ and $\mathcal{L}_2$ intersecting neither of them as shown on the left. We can move the voxel along the x axis toward $\mathcal{L}_2$ for a finite amount of distance until it hits $\mathcal{L}_2$ at one of its corners a. Then, there must exist another line $\mathcal{L}_1'$ parallel to both $\mathcal{L}_1$ and $\mathcal{L}_2$ passing through corner b of the voxel, which is diagonal to a. The distance d' between $\mathcal{L}_1'$ and $\mathcal{L}_2$ is: $d' = \sqrt{2}h \cdot \sin(\theta + \pi/4) = h \cdot (\sin\theta + \cos\theta) = \widehat{d}$. Since $\mathcal{L}_1'$ lies between $\mathcal{L}_1$ and $\mathcal{L}_2$, we have $d > d'$, which contradicts our assumption. ■

Theorem 2.4.1 suggests that as long as the scanning condition is satisfied, no missing voxels will be produced, and the resulting voxelization is guaranteed to be a cover of $\mathcal{T}_i$. On the other hand, if the scanline interval exceeds $\widehat{d}$, the resulting voxelization $\mathcal{V}_i$ is no longer 26-tunnel-free, and we can clearly see voxels missed as shown in Fig. 2.5. The voxelization of the entire triangle $\mathcal{V}$, is the union of the voxelization of each $\mathcal{T}_i$: $\mathcal{V} = \bigcup \mathcal{V}_i$. Because $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$ always share a base edge, $\mathcal{V}_i$ overlaps $\mathcal{V}_{i+1}$ at voxels corresponding to this shared edge. Thus, $\mathcal{V}_i \bigcup \mathcal{V}_{i+1}$ is also 26-tunnel-free and so is $\mathcal{V}$.

Integer scanline While the parallel scanline method is theoretically optimal, it needs to compute its intersections between the side edges of $\mathcal{T}_i$ for each scanline in

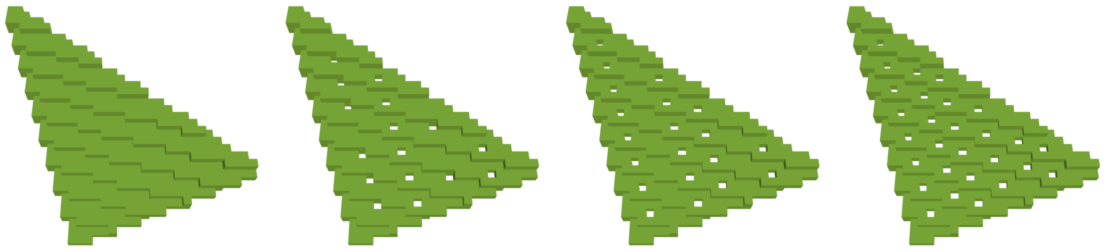


Figure 2.5: From left to right: the resulting voxelization of a triangle with scanline intervals of $\widehat{d}$, $1.05\widehat{d}$, $1.1\widehat{d}$ and $1.2\widehat{d}$.

order to determine the starting and ending locations of the scanline voxelization. The previous voxelization of $\mathcal{T}$'s edges is completely disregarded. In addition, floating number arithmetic is unavoidable as the optimal interval $\widehat{d}$ itself is a floating number. Similar to ILV, we further tweak the parallel scanline algorithm and provide its integer-only counterpart, which requires only an incremental integer arithmetic at each step.

Let $\mathcal{V}_A$ and $\mathcal{V}_B$ be the voxelizations of two side edges e_A and e_B of $\mathcal{T}_i$, which contain two sets of ordered voxels along e_A and e_B respectively[2]. We restrict the starting and ending points, referred as end A and end B, of a scanline to be the grid points in $\mathcal{V}_A$ and $\mathcal{V}_B$. Such constraint frees us from expensive calculations for finding the exact intersections between a scanline and side edges – we only need to identify the best scanline ends from sets $\mathcal{V}_A$ and $\mathcal{V}_B$.

Undoubtedly, the first scanline connects the first entries in $\mathcal{V}_A$ and $\mathcal{V}_B$, say $\mathbf{P}_A(X_A, Y_A) \in \mathcal{V}_A$ and $\mathbf{P}_B(X_B, Y_B) \in \mathcal{V}_B$ as shown in Fig. 2.6. According to Eq. (2.5), we re-write the best SI as:

$$\widehat{d} = h \cdot (\sin\theta + \cos\theta) = \frac{|\Delta X_{AB}| + |\Delta Y_{AB}|}{\sqrt{\Delta X_{AB}^2 + \Delta Y_{AB}^2}}, \tag{2.6}$$

where $\Delta X_{AB} = X_A - X_B$ and $\Delta Y_{AB} = Y_A - Y_B$. The line equation of the current

[2]There will be three side edges if $\mathcal{T}_i$ is a pentagon. In this case, we simply combine the two connected side edges into one single side edge.

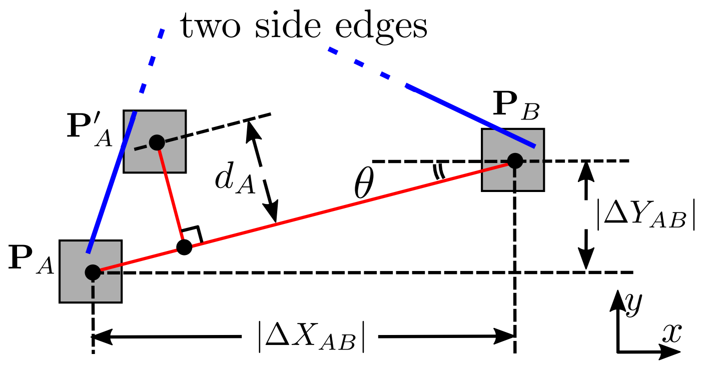


Figure 2.6: We find the next voxel $\mathbf{P}'_A \in \mathcal{V}_A$ as the one that is most distant from the current scanline while the scanning condition is still satisfied.

scanline can be written in the form of $ax + by + c = 0$, where $a = \Delta Y_{AB}$, $b = \Delta X_{AB}$, $c = \Delta X_{AB} Y_A - \Delta Y_{AB} X_A$. It is known that the distance between a point (x_0, y_0) and $ax+by+c = 0$ is $|ax_0+by_0+c|/\sqrt{a^2+b^2}$. Thus, d_A, between a voxel $\mathbf{P}'_A(X'_A, Y'_A) \in \mathcal{V}_A$ and the current scanline can be computed as:

$$d_A = \frac{1}{\sqrt{\Delta X_{AB}^2 + \Delta Y_{AB}^2}} \Big| \Delta Y_{AB}(X'_A - X_A) - \Delta X_{AB}(Y'_A - Y_A) \Big|. \tag{2.7}$$

Enforcing $d_A \leq \widehat{d}$ leads to $\Big|\Delta Y_{AB}(X'_A - X_A) - \Delta X_{AB}(Y'_A - Y_A)\Big| \leq |\Delta X_{AB}| + |\Delta Y_{AB}|$ or equivalently:

$$C_1 \leq \Delta Y_{AB} X'_A - \Delta X_{AB} Y'_A \leq C_2. \tag{2.8}$$

Here, $C_1 = \Delta Y_{AB} X_A - \Delta X_{AB} Y_A - |\Delta X_{AB}| - |\Delta Y_{AB}|$ and $C_2 = \Delta Y_{AB} X_A - \Delta X_{AB} Y_A + |\Delta X_{AB}| + |\Delta Y_{AB}|$. Every time we update $\mathbf{P}'_A$ with its next entry in $\mathcal{V}_A$, either its X or Y indices will be changed by ± 1, meaning Eq. (2.8) can actually be incrementally evaluated:

$$C_1 \leq C_0 + \Delta C \leq C_2, \qquad C_0 = \Delta Y_{AB} X_A - \Delta X_{AB} Y_A, \tag{2.9}$$

where ΔC could be either $\pm\Delta Y_{AB}$ or $\pm\Delta X_{AB}$, depending on the orientation of e_A.

Note that C_0, C_1 and C_2 are all integer constants depending on the current scanline configuration, and we can tell whether $\mathbf{P}'_A$ violates the scan condition with only

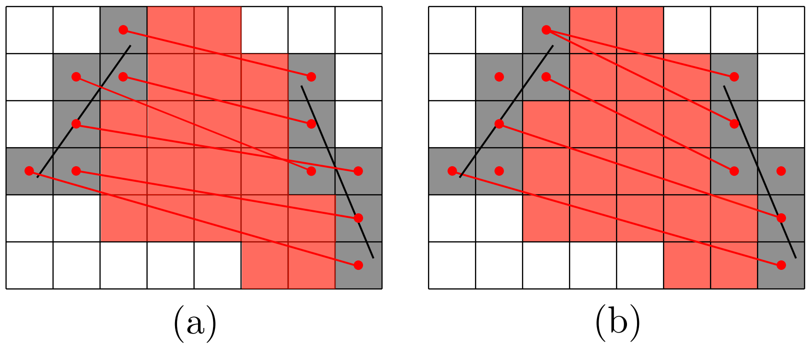
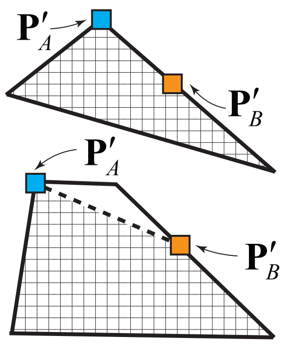


Figure 2.7: Compared to the naïve scanline strategy (a), the proposed method skips most redundant voxel generation (b).

four integer operations: three comparisons (one for determining the value of ΔC) and one addition. As soon as the scan condition does not hold, we rollback to the previous entry in $\mathcal{V}_A$ and choose it as the end A for the next scanline. Otherwise, current $\mathbf{P}'_A$ may still be conservative and we forward to the next entry in $\mathcal{V}_A$. Similarly, we can locate the best end B in $\mathcal{V}_B$, and voxelize the resulting scanline using ILV. Fig. 2.7 shows an illustrative example of the proposed scanline method, as well as the naïve scanline strategy, in which scanlines are generated for every voxel pair in $\mathcal{V}_A$ and $\mathcal{V}_B$. Experiments show that our method is significantly faster than the naïve scanline since most of the redundant voxel generation is omitted. The complete triangle voxelization procedure is summarized in Alg. 1.



Boundary treatment It is noteworthy that the numbers of voxels in $\mathcal{V}_A$ and $\mathcal{V}_B$ may differ significantly and it is possible that, for example as shown on the right, $\mathbf{P}'_A$ reaches the last entry in $\mathcal{V}_A$ before $\mathbf{P}'_B$ does. Such boundary inconsistency is handled seperately based on the geometry of $\mathcal{T}_i$. If $\mathcal{T}_i$ is a triangle, the scanline terminates as soon as $\mathbf{P}'_A$ (or $\mathbf{P}'_B$) reaches the end because the entire triangle interior has been voxelized (the shadowed region). Otherwise, the corresponding base edge will

Algorithm 1 Triangle Voxelization

```
1: function VoxelizeTriangle($\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{n}$)
2:     $\{\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2\} \leftarrow$ GetVoxel($\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$)
3:     $i \leftarrow$ DominantAxisIndex($\mathbf{n}$)
4:     SortOnAxis($\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, i$)
5:     MarkLineILV($\mathbf{P}_0, \mathbf{P}_1, \mathcal{Q}_0$)
6:     MarkLineILV($\mathbf{P}_1, \mathbf{P}_2, \mathcal{Q}_1$)
7:     MarkLineILV($\mathbf{P}_0, \mathbf{P}_2, \mathcal{Q}_2$)
8:     $\mathcal{Q}_1 \leftarrow \mathcal{Q}_0 \cup \mathcal{Q}_1$
9:     FillInterior($\mathcal{Q}_1, \mathcal{Q}_2, \mathbf{P}_0, \mathbf{P}_2, i$)
10: ________________________________________
11: function MarkLineILV($\mathbf{P}_0, \mathbf{P}_1, \mathcal{Q}$)
12:     $\Delta\mathbf{P}[0] \leftarrow$ Sign($\mathbf{P}_1[0] - \mathbf{P}_0[0]$)
13:     $\Delta\mathbf{P}[1] \leftarrow$ Sign($\mathbf{P}_1[1] - \mathbf{P}_0[1]$)
14:     $\Delta\mathbf{P}[2] \leftarrow$ Sign($\mathbf{P}_1[2] - \mathbf{P}_0[2]$)
15:     $\mathbf{L}[0] \leftarrow \mathbf{M}[0] \leftarrow |\mathbf{P}_1[1] - \mathbf{P}_0[1]||\mathbf{P}_1[2] - \mathbf{P}_0[2]|$
16:     $\mathbf{L}[1] \leftarrow \mathbf{M}[1] \leftarrow |\mathbf{P}_1[0] - \mathbf{P}_0[0]||\mathbf{P}_1[2] - \mathbf{P}_0[2]|$
17:     $\mathbf{L}[2] \leftarrow \mathbf{M}[2] \leftarrow |\mathbf{P}_1[0] - \mathbf{P}_0[0]||\mathbf{P}_1[1] - \mathbf{P}_0[1]|$
18:     $\mathbf{P}_{current} \leftarrow \mathbf{P}_0$
19:     while $\mathbf{P}_{current} \neq \mathbf{P}_1$ do
20:         $\langle \mathbf{L}_{min}, L_{index} \rangle \leftarrow$ Min($\mathbf{L}[0], \mathbf{L}[1], \mathbf{L}[2]$)
21:         $\mathbf{P}_{current}[L_{index}] \leftarrow \mathbf{P}_{current}[L_{index}] + \Delta\mathbf{P}[L_{index}]$
22:         $\mathbf{L} \leftarrow \mathbf{L} - \mathbf{L}_{min}$
23:         $\mathbf{L}[L_{index}] \leftarrow 2\mathbf{M}[L_{index}]$
24:         MarkVoxel($\mathbf{P}_{current}$)
25:         $\mathcal{Q}$.PushBack($\mathbf{P}_{current}$)
26: ________________________________________
27: function FillInterior($\mathcal{Q}_1, \mathcal{Q}_2, \mathbf{P}_0, \mathbf{P}_2, axis$)
28:     for $i = 0$ to $\mathbf{P}_2[axis] - \mathbf{P}_0[axis]$ do
29:         $slice \leftarrow \mathbf{P}_0[axis] + i + 1/2$
30:         $\mathcal{Q}_{1sub} \leftarrow$ GetSubSequence($\mathcal{Q}_1, slice$)
31:         $\mathcal{Q}_{2sub} \leftarrow$ GetSubSequence($\mathcal{Q}_2, slice$)
32:         while $\mathcal{Q}_{1sub} \neq \emptyset \mathbf{OR} \mathcal{Q}_{2sub} \neq \emptyset$ do
33:             $\mathbf{P}_{start} \leftarrow$ GetNextInSlice($\mathcal{Q}_{1sub}$)
34:             $\mathbf{P}_{stop} \leftarrow$ GetNextInSlice($\mathcal{Q}_{2sub}$)
35:             MarkLineILV($\mathbf{P}_{start}, \mathbf{P}_{stop}$)
```

become the new side edge, and the remaining un-voxelized region becomes a triangle.

2.5 Performance Analysis

In this section, we discuss the performance of the proposed voxelization algorithm. We also elaborate the detailed algorithmic difference between our method and existing SAT-based techniques.

Cover property The cover property of our resulting surface voxelization is determined by which line voxelization algorithm, SLV, RLV or ILV, is used. It is not difficult to see that using SLV can form the super cover of the input model as the state-of-the-art does. When using RLV, the proposed method forms a 26-tunnel-free cover. As discussed previously, if the input model does not contain singular points, both RLV and SLV generate the super cover. ILV produces 26-tunnel-free surface voxelization which is a close approximation of a cover. The error induced by ILV is related to many factors, such as the voxel dimensions, the orientation of the line segment and the relative spatial position between the grid points and the line end points. Fig. 2.8 shows three typical cases where we use RLV and ILV to voxelize some line segments. We can see that the voxelizations vary a lot in the extreme cases (a) and (b), but in (c), which is like the average case, no error occurred.

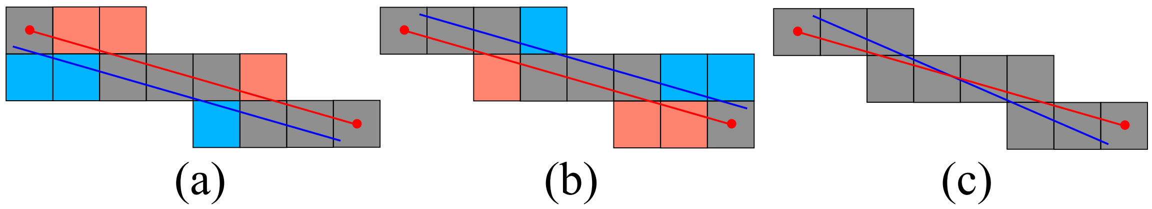


Figure 2.8: Voxelizing the line segments (blue) by RLV and the approximation (red) by ILV could result either different voxelizations in (a) and (b), or identical voxelizations in (c). The grey voxels are captured by both RLV and ILV, while the blue voxels and the red voxels are captured only by RLV or ILV respectively.

Optionally, our method can be downgraded to generate thinner 18- or 6-tunnel-free voxelizations by removing some base edge voxelizations (Fig. 2.9). Recall that the base edges are shared by two consecutive polygons and voxelized in both of the corresponding two slices. This feature is necessary to make the final voxliezation 26-tunnel-free. If we eliminate the voxelization of the base edge in either of the two slices, 26- or 18-paths form, and the voxelization can only be 6-tunnel-free and is not a cover anymore. The downgrading method makes thinner voxelizations as preferred in some applications.

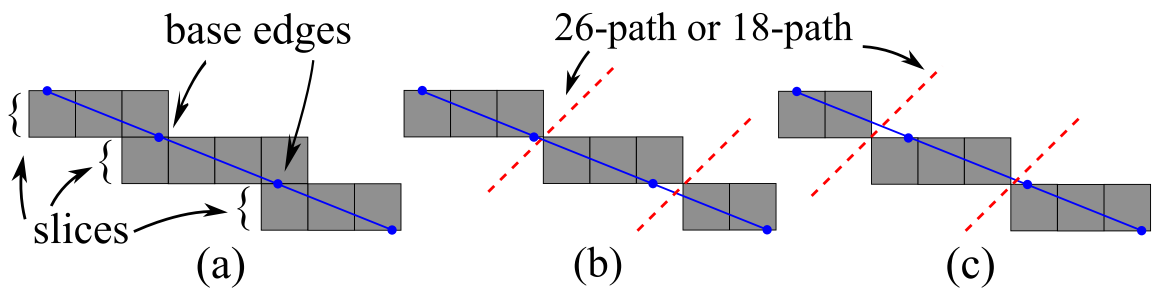


Figure 2.9: Our method (a) can be downgraded to 18- or 6-tunnel-free methods (b) or (c) by eliminating some base edge voxelizations. The resulting voxelizations are thinner and 6-tunnel-free.

Algorithmic complexity Schwarz and Seidel [SS10] proposed the state-of-the-art voxelization algorithm. They firstly determine the dominate axis of the triangle, say, z axis, then project the triangle to the xy plane. A set of voxels are determined by a 2D overlap test. Each voxel further determines a voxel array along z axis which contains a number of voxel candidates (up to three) intersecting the triangle. All the voxel candidates are subject to two remaining 2D overlap tests. The complexity of the method is therefore $\mathbf{O}(N \times M)$, where N and M are the numbers of voxels along x and y axis of the bounding box respectively. We define a triangle-voxel overlap test as an *atomic operation*, which contains 9 sub-tests. Each sub-test has 2 multiplications, 2 additions and 1 comparisons. All of the $N \times M$ voxels should undergo the atomic operation, but most of them do not have to take all the sub-tests.

There are two reasons. One is that a failed sub-test will imply no overlap and the rest sub-tests are not necessary. The other is that voxels in the same voxel array can share the xy plane test result. Therefore, based on the observation that the thickness of the voxelization is at most three voxels along z axis, the lower bound of the number of sub-tests for an overlapping voxel is 7, which yields 14 multiplications, 14 additions and 7 comparisons. Pantaleoni [Pan11] further improved the previous method when determining the voxel candidates in the xy plane. This method takes one coordinate as a constant and computes the range of the other coordinate through the edge functions. The voxels in the range are immediately taken as candidates without performing overlap tests. The rest steps are the same as the ones in the previous method: the z coordinate range is computed per voxel array, and voxels in the range are subjected to the remaining two 2D projection overlap tests. We can see that the complexity of this method is $\mathbf{O}(W)$, where W is the number of the voxles overlapping the triangle. For an overlapping voxel, 6 sub-tests are needed, which contains 12 multiplications, 12 additions and 6 comparisons.

In our method, we determine the dominant axis, followed by voxlelizing the three edges of the triangle. The rest work is to find the voxels overlapping the triangle interior. To this end, we further divide the triangle slice by slice. In each slice, we use 2D scanlines to find the overlapping voxels. We can see that the method actually compute indices of the overlapping voxels only. Therefore, the complexity of our method is also $\mathbf{O}(W)$. However, the atomic operation of our method is simply to extend one voxel along a 2D scanline, which only involves 2 additions plus 1 comparison. This is the major reason why our method is faster than the existing SAT-based method.

Parallelization Both aforementioned SAT-based methods and our method can be parallelized and accelerated using multi-threading or GPU. However, it can be clearly seen that our method is parallelized for each triangle, while the SAT-based

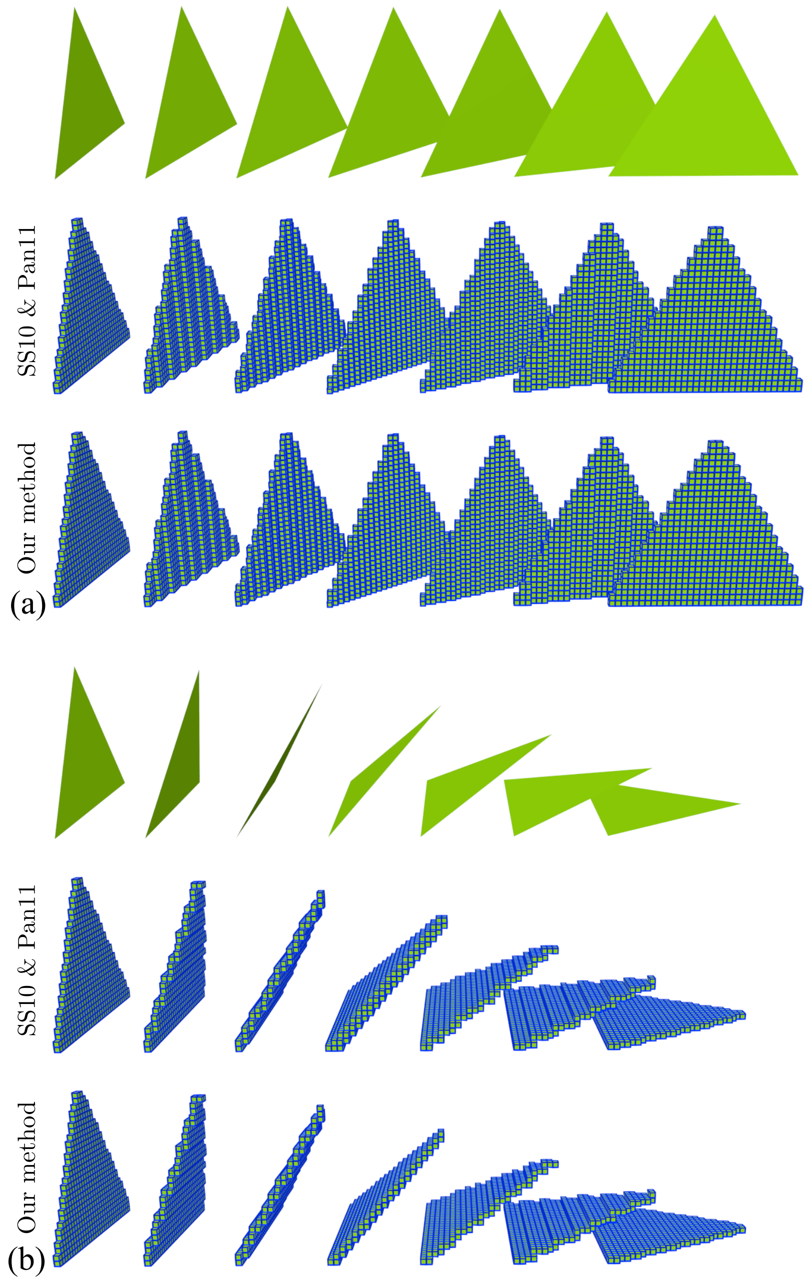


Figure 2.10: Voxelizing an equilateral triangle under the resolution of 32^3. (a) The triangle rotate around y axis. (b) The triangle rotates around z axis.

methods can be trivially parallelized for each voxel candidate. As a result, one may speculate that with the increased resolution of the vocalization, SAT-based method will eventually outperform our method. Interestingly, the fact is opposite – our method often demonstrates a better performance in practice. The reasons are two-fold. First of all, while scanning the interior of an individual triangle is *sequential*, parallelization at triangles utilizes modern GPU platform already. For instance, the latest `nVidia GTX 1080` GPU equips with $2,560$ cores, while most meshes we are dealing with nowadays have much more than $2,560$ triangles in general. More importantly, our method has a much simpler atomic operation than SAT-based methods (e.g. in [SS10, Pan11]). Therefore, a higher voxelization resolution will further exaggerate such difference (see Tab. 2.2). On the other hand, if the dimension of a voxel is comparable to a triangle patch, our method becomes less efficient. This is because interior voxels may be already generated during the line voxelization or even point voxelization stage and computing the SI based on Eq. (2.8) is less profitable to reduce the redundant voxelization. Furthermore, if the triangle numbers are very small (e.g. only one triangle), the SAT-based methods will beat our method easily by parallelization. Based on such analysis, one can clearly see that our method is alternative to and complements existing SAT-based methods: it is suitable for high-density voxelization (e.g. for accurate numerical integrations). Our experiment results confirm this conclusion.

2.6 Experiments and Results

Our method is tested on a desktop PC (with both CPU and GPU), and an Apple iPhone 6. For comparison, we also implemented the state-of-the-art SAT-based methods as in [SS10] and [Pan11]. The desktop PC equips an Intel$^{®}$ i7-2600 CPU@3.4 GHz (4 physical cores), 12G RAM, and a NVIDIA GTX 970 video card. The mobile platform is an Apple iPhone 6 with Dual-core Typhoon CPU@1.4 GHz (ARM v8-based). Our CPU implementation is both single- and multi-thread using `C++` and our GPU implementation utilizes NVIDIA `CUDA 7.5`. Our IOS implementation is

y	# Comparisons			# Additions		
	SS10	**Pan11**	**Ours**	**SS10**	**Pan11**	**Ours**
0°	2,483	0	838	4,966	0	1,111
15°	6,356	3,806	863	12,712	7,612	1,236
30°	8,177	5,343	1,061	16,354	10,686	1,687
45°	7,835	5,523	1,027	15,666	11,046	1,744
60°	8,151	5,246	1,068	16,302	10,492	1,701
75°	6,357	3,786	863	12,750	7,572	1,236
90°	2,483	0	839	4,966	0	1,113
z	**SS10**	**Pan11**	**Ours**	**SS10**	**Pan11**	**Ours**
0°	2,483	0	838	4,966	0	1,111
15°	5,899	3,490	863	11,798	6,980	1,236
30°	6,204	3,940	1,061	12,408	7,880	1,687
45°	8,162	5,358	1,027	16,324	10,716	1,744
60°	6,204	3,988	1,068	12,408	7,976	1,701
75°	5,899	3,502	863	11,798	7,004	1,236
90°	2,483	0	839	4,966	0	1,113

Table 2.1: Comparative statistics of complexity of SAT-based methods ([SS10] and [Pan11]) and our method showing the total numbers of comparisons and additions during the voxelization. When the triangle is parallel to the coordinate plane, no atomic operation is needed in [Pan11].

written in `Object-C` with `Xcode`.

Implementation In [SS10] and [Pan11], triangles can be preprocessed and classified to 1D, 2D or 3D cases. In 1D cases, the triangle is slim and enclosed in one voxel array aligned to a coordinate axis. All the voxels in the array are marked as overlapping. In 2D cases, each voxel in the bounding box undergoes a 2D projection overlap test. Voxels passing the test are marked immediately [SS10]. More efficiently, Pantaleoni [Pan11] finds those overlapping voxels by fixing one coordinate and computing the range of the other coordinate. In more general 3D cases, as described previously, one projects the triangle to determine the set of voxel arrays, computes the range of the voxel candidates and processes the two remaining 2D projection overlap tests.

In our CUDA implementation we process all triangles in parallel, with each thread voxelizing a single triangle. Each voxel in our global grid is represented as a bit in a 32-bit integer array and each thread that is processing a triangle must update this array when a voxel is labeled. To accomplish this we take advantage of `atomic functions` from CUDA. Atomic memory operations alleviate the complexity involved in updating shared memory during a parallel computation. Specifically, we utilize the atomic OR function which ensures that all voxel array updates issued concurrently are performed without interruption with respect to other threads.

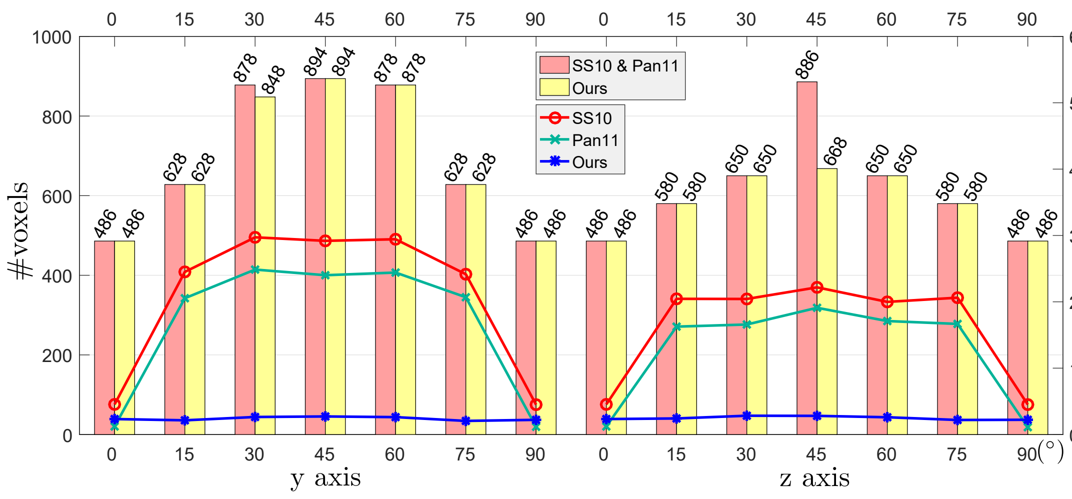


Figure 2.11: The numbers of voxels (bar plots) as well as the time performance (line plots) of [SS10] [Pan11] and our method.

Voxelization of a single triangle We voxelize an equilateral triangle under the resolution of 32^3 using the SAT-based methods and our method. The triangle is initially aligned with the yz plane, and we rotate it around y and z axes gradually by 15° each time up to 90°. We use the single-thread implementation in this experiment. The resulting voxelizations are shown in Fig. 2.10. The corresponding computation costs are reported in Tab. 2.1, where we can see that our method invests much less computation in the general 3D scenarios. Note that we don't compare the multipli-

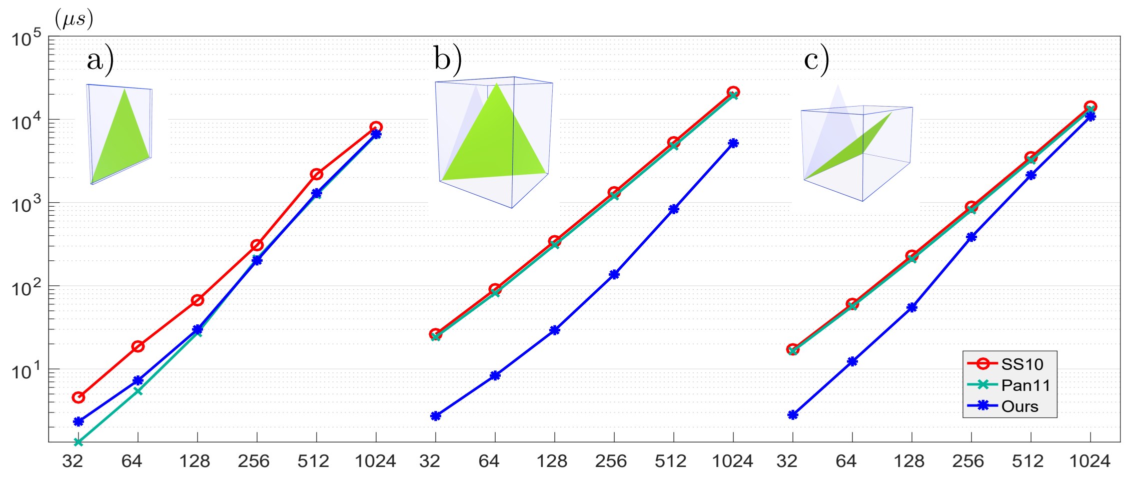


Figure 2.12: The time performance of triangle voxelization under various resolutions: a) the triangle is aligned in the yz plane initially; b) the triangle is rotated around y axis by 45°; and c) the triangle is rotated around z axis by 45°.

cations because the atomic operation in our method does not involve multiplications.

We also notice that the computation costs of the SAT-based method largely depend on the orientation of the triangle, while our method is much more consistent with respect to different orientations. This observation is also verified in the time benchmark shown in Fig. 2.11, where we pick three typical poses of the rotating triangle, which are the initial one, the one rotated around y axis by 45°, and the one rotated around z axis by 45°, to test the time performance under various resolutions from 32^3 to 1024^3 as shown in Fig. 2.12. In Fig. 2.12 a), the triangle is essentially in 2D and [Pan11] performs best. However, in more general 3D cases (Fig. 2.12 b) and c)) our method is faster. Note that the vertical axis is in logarithmic.

More results on 3D models The voxelization results of more 3D models are shown in Fig. 2.13. The resolutions of the voxelization increases from 256^3 to 4096^3. Tab. 2.2 reports the comparative time performance using a single- and multi-thread CPU, `CUDA` and iOS implementations (i.e. Fig. 2.14) of both the SAT-based methods and our method. Due to the limited memory, voxelization either at 2048^3 on the

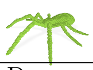
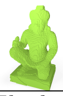
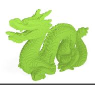
Model	**Reso.**	**CPU**^{s} (ms)			**CPU**^{m} (ms)			**GPU** (ms)			**iOS** (ms)		
		Pan11	**Ours**	**Acc**	**Pan11**	**Ours**	**Acc**	**Pan11**	**Ours**	**Acc**	**Pan11**	**Ours**	**Acc**
Spider	256^3	3.7	2.7	37%	1.8	1.6	13%	0.30	0.20	50%	8.5	7.4	15%
(3,341 tris)	512^3	8.8	4.6	91%	3.1	2.5	24%	0.85	0.45	88%	20.1	10.9	84%
	1024^3	24.8	9.7	156%	8.8	4.7	87%	2.6	1.1	132%	83.3	29.0	187%
	2048^3	87.9	29.9	194%	37.0	17.8	107%	20.7	8.6	139%	–	–	–
	4096^3	492.3	187.7	162%	148.8	62.8	137%	–	–	–	–	–	–
Demon	256^3	11.1	8.2	35%	5.3	3.9	36%	0.82	0.63	30%	21.5	16.7	29%
(8,822 tris)	512^3	30.5	14.5	110%	13.2	5.6	136%	1.8	1.5	20%	83.6	36.0	132%
	1024^3	99.2	33.3	198%	29.7	12.1	145%	6.8	4.2	62%	284.2	99.2	186%
	2048^3	476.1	117.0	307%	144.6	48.3	199%	33.1	19.5	70%	–	–	–
	4096^3	2255	759.8	197%	699.4	241.8	189%	–	–	–	–	–	–
Elephant	256^3	11.0	7.9	39%	4.8	4.2	14%	1.9	1.8	6%	21.9	16.2	35%
(10,150 tris)	512^3	28.4	14.1	101%	9.0	6.9	30%	4.1	2.3	78%	79.9	36.0	122%
	1024^3	86.0	31.3	175%	26.7	14.8	80%	15.4	4.6	234%	380.5	95.2	300%
	2048^3	325.2	103.9	213%	132.3	53.2	148%	85.4	22.7	276%	–	–	–
	4096^3	2066	711.3	190%	594.6	216.6	175%	–	–	–	–	–	–
Sailboat	256^3	23.4	19.2	22%	14.8	13.7	8%	10.6	4.3	66%	62.0	32.8	89%
(70,476 tris)	512^3	49.0	29.1	69%	25.7	18.0	43%	17.9	9.7	85%	197.4	56.9	247%
	1024^3	133.9	50.9	163%	58.8	27.9	111%	48.8	22.9	113%	491.5	110.2	346%
	2048^3	483.4	104.9	361%	173.4	49.3	252%	160.8	42.1	281%	–	–	–
	4096^3	1168	333.9	250%	394.5	121.3	225%	–	–	–	–	–	–
Dragon	256^3	59.3	61.1	−3%	35.1	38.2	−8%	1.3	1.4	−6%	72.3	80.1	11%
(100,000 tris)	512^3	99.4	85.9	16%	48.7	47.2	3%	1.7	1.6	6%	191.2	143.4	33%
	1024^3	197.6	128.3	54%	83.3	62.5	33%	6.5	5.2	13%	618.0	302.4	104%
	2048^3	591.8	253.0	134%	191.5	103.9	84%	22.8	17.5	30%	–	–	–
	4096^3	2507	1007	149%	629.5	309.7	103%	–	–	–	–	–	–

Table 2.2: The running time for GPU/CPU and iOS implementations of [Pan11] (**Pan11**) and our method. The accelerations (**Acc**) are also highlighted. **CPU**^{s} and **CPU**^{m} are for the single- and multi-thread implementations. The number of threads that we use in the latter is four.

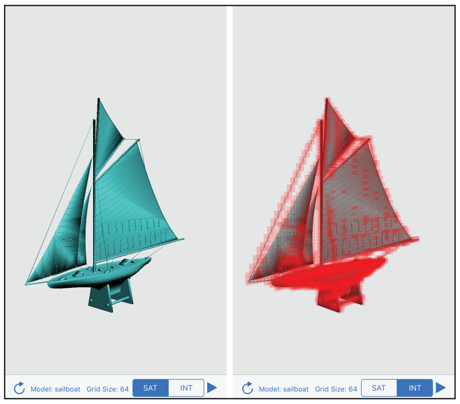


Figure 2.14: Snapshots of our iOS implementation.

iPhone or at 4096^3 on GPU is not available. As Tab. 2.2 exhibits, the performance gap between the SAT and our method is getting larger with the increased voxel resolutions. Such result is consistent with our performance analysis as discussed in Sec. 2.5. The performance reported in Tab. 2.2 is based on the integer version of our algorithm. We find that the acceleration of using integer version algorithm over the floating number version is very minor ($\sim 3\%$). However, further acceleration

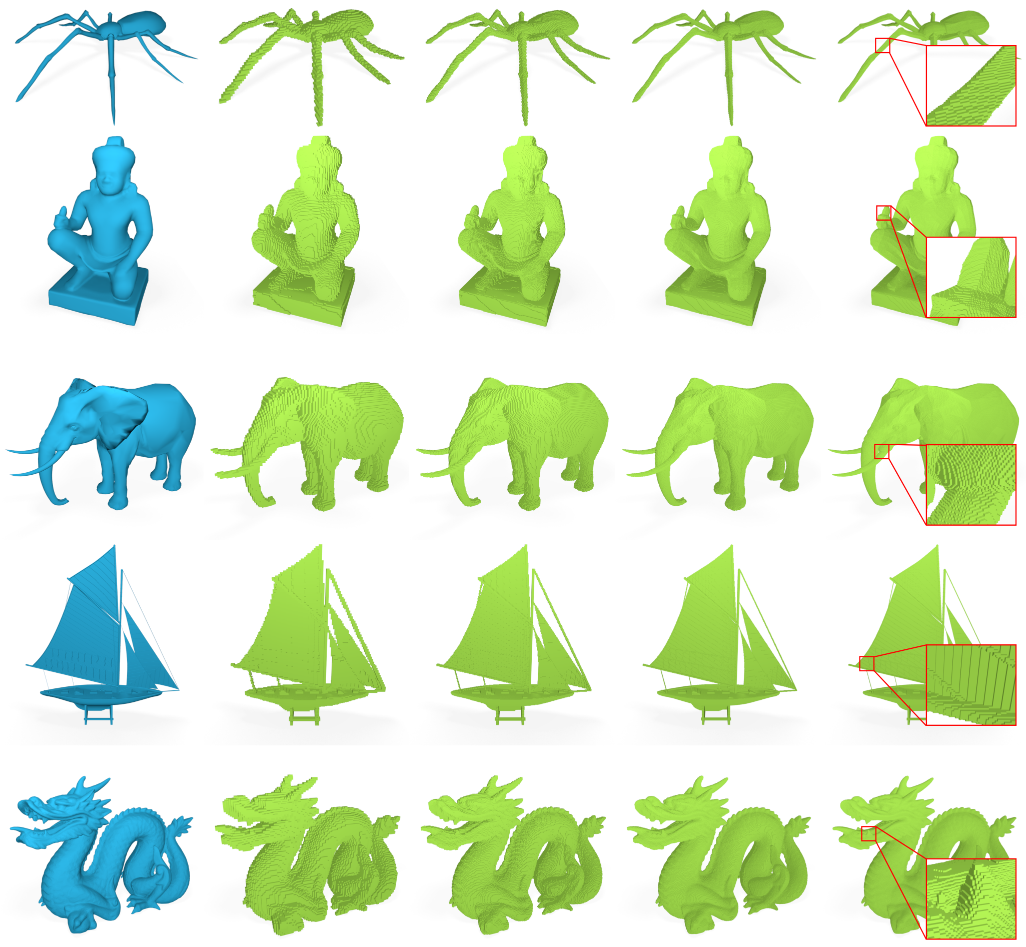


Figure 2.13: Voxelizing several 3D models under the voxelizations of 128^3, 256^3, 512^3 and 1024^3.

of using integer version algorithm may be possible
if dedicated hardware is adopted [LM00].

To justify the voxelization error, we voxelize the 3D models by different methods and compare the voxel numbers of the resulting voxelizations. The results are reported in Tab. 2.3. We can see that, as expected, our floating number version (SLV/RLV) generates the same number of voxels as the SAT-based method does, and the relative error induced by our integer version is small (less than 2.5%). We highlight the difference of two voxelizations generated by the SAT-based method and

3D Model	Spider			Demon			Elephant			Sailboat			Dragon		
Resolution	256^3	512^3	1024^3	256^3	512^3	1024^3	256^3	512^3	1024^3	256^3	512^3	1024^3	256^3	512^3	1024^3
SS10/Pan11	38.9	161	658	173	691	2.77	125	506	2.05	65.6	276	1.18	168	673	2.70
SLV/RLV	38.9	161	658	173	691	2.77	125	506	2.05	65.6	276	1.18	168	673	2.70
ILV	39.8	164	674	175	701	2.80	128	516	2.08	67.2	281	1.20	171	687	2.75
ILV error	2.3%	1.9%	2.4%	1.2%	1.4%	0.7%	2.4%	2.0%	1.5%	2.4%	1.8%	1.7%	1.8%	2.1%	1.9%

Table 2.3: Comparative statistics of the voxel numbers of the resulting 3D model voxelizations under different resolutions and using different methods. The data in blue color are in million voxels, while others are in kilo voxels. Our floating number version (SLV/RLV) generates the same number of voxels as the SAT-based method does. The relative error induced by our integer version method (ILV) is less than 2.5%.

our integer version method respectively in Fig. 2.15 (under the 64^3 resolution).

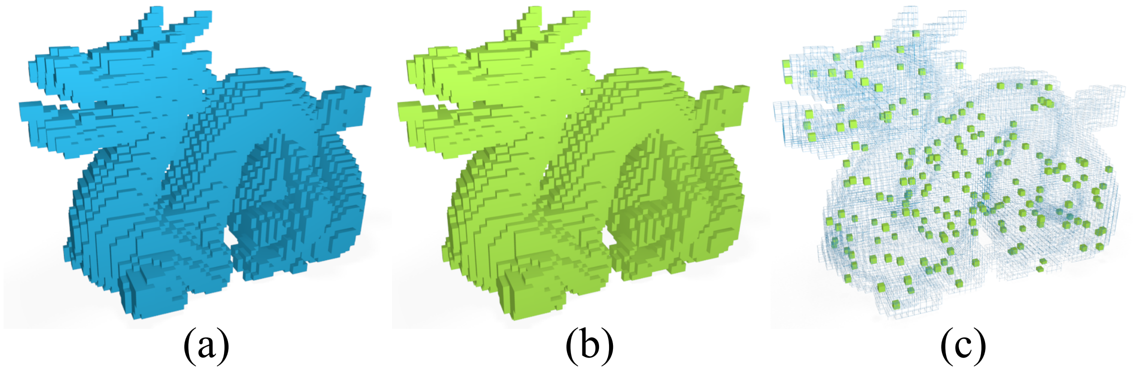
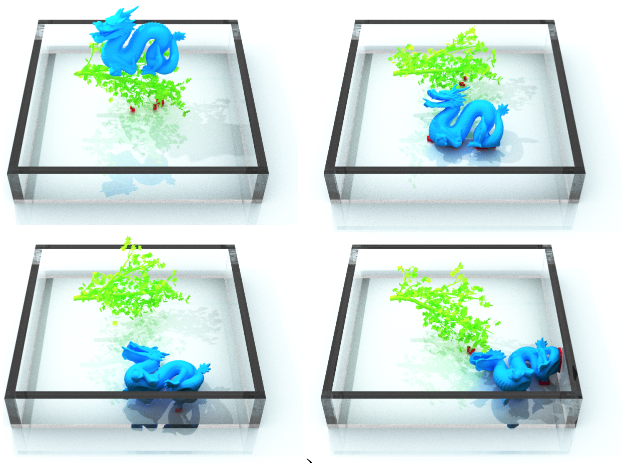
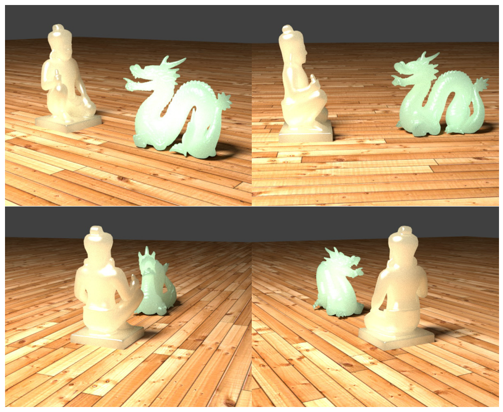


Figure 2.15: Voxelizations of the 3D dragon model by the SAT-based method (a) and our integer version method (b) under the 64^3 resolution are presented. The difference is highlighted in (c).

Applications As mentioned, the voxelization plays an essential role in many graphics applications. The proposed high-performance voxelization algorithm can be directly used in physics-based animations – both for mesh generation and collision culling. It is also important for global illumination (Fig. 2.16).



(a)



(b)

Figure 2.16: Our methods are used in mesh generation and collision culling (a) and global radiosity (b).

2.7 Conclusions

We present a high-performance algorithm for the voxelization of complex 3D models. Our method avoids (relatively) expensive triangle-voxel intersection tests and voxelizes the surface geometry based on an efficient line voxelization algorithm. Since there is no optimal 3D scanline interval, we project a 3D triangle into axis-aligned slices and give the theoretically optimal scanline strategy. On top of it, we further avoid floating number arithmetic during the voxelization. We provide a comprehensive analysis and comparative experiments between the proposed method and the state-of-the-art SAT-based methods. The time performance on CPU, GPU as well as mobile devices shows that our method is efficient, especially for high-resolution voxelizations when the encapsulating a triangle requires more interior voxels.

Since the voxelization is a discrete representation of the original object, many applications require that voxels store more information rather than the binary overlap flag, such as density, normal vector, material properties, etc. For instance, surface normal vector at the voxel's location should be well estimated for alias-free rendering [WK93, SK99]; evaluating the occupancy functions (filtered techniques) [SK98] or distance functions (distance field techniques) [BW01, JBS06, PF01, NDS10] at the voxels is essential to reconstruct the original object surface. Our method is basically designed for binary voxelization and can not directly support the non-binary application. As a potential solution, one can first voxelize the object using our method, then compute and store the associated quantities for each overlapping voxel using other related algorithms.

Chapter 3

Autonomous Pavement Surface Recognition

3.1 Introduction

High quality pavement serviceability is critical to maintain safe and effective traffic operations. As an indispensable component of the Pavement Management Systems (PMS), the pavement condition evaluation is an essential procedure to provide comprehensive information for its serviceability quantification and maintenance scheduling [CZZ$^+$14]. It is generally composed of two major procedures: the pavement distress evaluation, which is conducted to calculate the Distress Rate (DR), and the pavement roughness assessment, which is performed to retrieve the International Roughness Index (IRI). State transportation agencies are responsible for examining pavement conditions within their jurisdiction on a regular basis and performing the road-way maintenance and rehabilitation accordingly. Generally, pavement condition information is collected through manual evaluations or automatic techniques. In a manual evaluation procedure, an inspector walking along roads visually evaluates the severity and extent of pavement distresses based on pre-specified criteria [BHM+12]. However, manual evaluation is labor-intensive and time-consuming, and the inspector is often at high risk of being in an accident even with preventive safety measurements. With these disadvantages in mind, automatic pavement detection techniques have been developed and gained increasing popularity among state transportation agencies. Automated pavement condition data are generally collected with automated and dedicated devices, such as pavement scan vans or aerial photo cameras. However, regardless of the data collection procedures, the quality of the data collected is always compromised to some extent due to the individual subjectivity in evaluating the severity and extent of pavement distresses [D+14, BMC10, UKCL10]. Therefore, computer-aided pavement distress detection and surface reconstruction methods are needed to minimize the impacts of human subjectivity in pavement condition distress assessments.

Considerable research has been conducted to assist pavement condition evaluation in using computer-aided techniques. For example, Tremblais and Augereau [TA04] proposed a fast multi-scale edge detection algorithm to detect pavement cracks. Bray et al. [BVLH06] proposed a neural network-based technique for an automatic classification of pavement cracks. Among all the existing computer-aided techniques, digital image processing is a mature method that has been increasingly utilized in pavement distress detection and road surface re-construction. Numerous studies have been conducted to improve the applicability and performance of image processing techniques for pavement surface evaluation. For instance, Mahler et al. [MKWS91] demonstrated the feasibility of using image processing techniques to detect cracks. Georgopoulos et al. [GLF95] developed an image processing techniques to automatically determine the type, extent, and severity of surface cracks for flexible road pavements. Although a wide range of algorithms have been developed to improve the performance of image processing techniques in pavement distress evaluation, most of these are based on 2D image information. Distress depth is not able to be measured directly but only inferred from overlapping 2D images. Therefore, estimation errors would be inevitably introduced and evaluation accuracy would be degraded. Ideally, width and length, are two measurements to evaluate pavement distress severity and extent, and depth is generally used to determine pavement maintenance and rehabilitation [LMG02]. Recent developments of 3D reconstruction approach enable a direct collection of 3D pavement distress information including not only width and but also the depth. 3D reconstruction relies on 3D point clouds (via inversely projecting the depth image pixels) collected by laser scanners or by stereo-vision algorithms-based video cameras [KB11]. In the past decades, significant efforts have been taken to investigate the applicability of 3D reconstruction techniques in pavement condition evaluation [LTD97,BBH+01,YSK+07]. For instance, Laurent et al. [LTD97] used an auto-synchronized laser scanning system to detect road rutting and cracking in high precision 3D environments. Other studies were also proposed to improve the performance 3D reconstruction techniques [LGG01,HTB03,HH09]. These studies provided comprehensive and in-depth understandings of pavement condition evaluations and pavement surface in 2D and 3D reconstructions. However, these techniques are either not maturely developed or too costly in practical applications, which impede their wider implementations.

Microsoft Kinect is an infrared-based sensory device enabling human-computer interaction without the assistance of any physical controllers. It operates by capturing user gestures. Kinect is able to produce real-time 3D surface data and has been widely applied in many fields, such as physical re-habilitation, education, cartography, etc. Tölgyessy and Hubinský [TH11] applied Kinect to robotics education, including data fusion, obstacle avoidance, collision detection, object recognition, gesture control, localization, and navigation. Compared to other aforementioned 3D reconstruction techniques, Kinect was originally developed for home entertainment and is very affordable at less than \$ 150 per unit. With its cost-effective and multi-disciplinary implementations, there is great potential to apply Kinect devices in pavement condition evaluations. This study is proposed to develop a cost-effective Kinect-based approach for 3D pavement surface reconstruction and cracking detection. Kinect fusion, point cloud conversion, mesh triangulation, and sharp feature examination modules are developed successively for crack recognition and severity identification. Human expert evaluation results are used as ground-truth data for comparison analyses. The results indicate that the proposed approach is able to reconstruct 3D surfaces, detect crack width, length, and depth information, and further identify distress severity levels based on the given protocols.

The rest of the paper is organized as follows: a comprehensive literature review is provided in Section 3.2. Section 3.3 introduces the Kinect fusion mechanism and the data collection procedure, followed by Section 3.4 which details the methodology we adopted. Section 3.5 discusses the experiment results and research limitations, and this research is concluded with Section 3.6.

3.2 Previous Work

Pavement surface distress information is essential in the pavement management program. Various levels of pavement maintenance activities and rehabilitation decisions are supported by pavement condition information [HHZ94]. Federal and State Departments of Transportation (DOTs) in the U.S. surveyed different types and numbers of distresses, and applied various pavement assessment approaches and pavement condition indices in their pavement evaluation procedures [FM09, BMC10, Gra, McG04]. For example at the federal level, the National Cooperative Highway Research Program (NCHRP) summarized existing data collection and processing techniques [McG04], as well as the data quality management issues and solutions [FM09] in automated pavement distress collection procedures. Meanwhile, at the state level, the Alabama Department of Transportation (ALDOT) utilizes manual evaluation methods in their pavement evaluation procedures. While manual surveys are still used among several states, the automated approaches have come into progressively more use. New Mexico Department of Transportation (NMDOT) applies both manual evaluation and automatic detection for pavement evaluations and uses a Pavement Serviceability Index (PSI) to measure pavement deteriorations. Oregon Department of Transportation (ODOT) applies automated data collection equipment for pavement evaluations. Considerable studies have also been performed to explore advanced techniques for pavement distress detection and pavement condition evaluations. For example, acoustic or laser sensors have been used to capture pavement cracking, aiming to relate cracking to abrupt variations in pavement texture [Gra]. Analog approach refers to the process wherein images are physically imposed on film or another median, like photographic and video [SFP96, WS11, GH93, MLP08]. The data captured by digital imaging approachs can be read electronically and processed or reproduced. Pavement surface reconstruction is a major procedure in automatic pavement evaluation analysis. Zhang and Elaksher [ZE12] developed image processing-based algorithms to quantify 3D details of pavement distresses using unmanned aerial vehicle (UAV) based image data. With a new image segmentation algorithm, Oh [Oh98] developed an image processing method to automatically analyze the recorded images and isolate distress features. Pynn et al. [PWL99] applied several new image processing algorithms to automatically detect the cracks by using video images collected with a van camera system. Pavement cracking, including longitudinal cracking, transverse cracking, alligator cracking and edge cracking, is a dominant category of pavement distress measurement, and the severity and extent of pavement cracking play significant roles in deteriorating the pavement serviceability. Therefore, a significant amount of research has been conducted to improve pavement cracking detection and measurement from different perspectives. Zhou et al. [ZHC05] proposed a wavelet-based image classification algorithm to detect cracks in pavement surfaces. Huang and Xu [HX06] presented an image processing algorithm customized for high-speed, real-time inspection of pavement cracking. Mustaffara et al. [MLP08] proposed a photogrammetry-based approach to automatically classify and quantify the pavement cracks. Ma [MZH08] proposed a method to detect cracks based on a non-subsampled contour transform algorithm. Oliveira and Correia [OC08] employed entropy and image dynamic thresholds to automatically segment road cracks. Chambon et al. [CGMN10] proposed to extract road cracks with adapted filtering and a Markov model-based segmentation. Distress depth information is an important contributing factor in determining pavement maintenance and rehabilitation [LMG02]. However, traditional pavement evaluation and surface reconstruction methods are not able to capture depth information directly and accurately. In the last two decades, along with the advances of 3D surface reconstruction techniques, distress depth detection, especially crack depth detection, became feasible. 3D surface reconstruction relies on 3D point clouds collected by laser scanners or by stereo-vision algorithms using a multiple calibrated cameras [KB11]. Microsoft Kinect is an infrared-based motion sensor that is able to gather real-time 3D geometric features, colors, and audio data of the environment [SLPR13]. With the merits of its mature techniques and affordable expenses, Kinect has been applied in many fields. Chang et al. [CLZ$^+$12] examined the application of Kinect devices in physical rehabilitation and found that they can provide competitive motion tracking performance in the comparison to other professional motion detection systems. Lange et al. [LKM$^+$12] investigated the interactive game-based rehabilitation using Kinect devices and proved their applicability in clinical use. Kitsunezaki et al. [KAMM13] performed a study of using Kinect for physical rehabilitation. Other research investigated the application of Kinect in education [Hsu11]. Ren et al. [RMY11] investigated the application of Kinect hand gesture recognition function in human-computer interactions. Kondori et al. [KYL$^+$11] studied the 3D head pose estimation using Kinect. Khoshelham and Elberink [KE12] studied the application of Kinect's depth data in the indoor mapping. Oliver et al. [OKWM12] investigated the application of

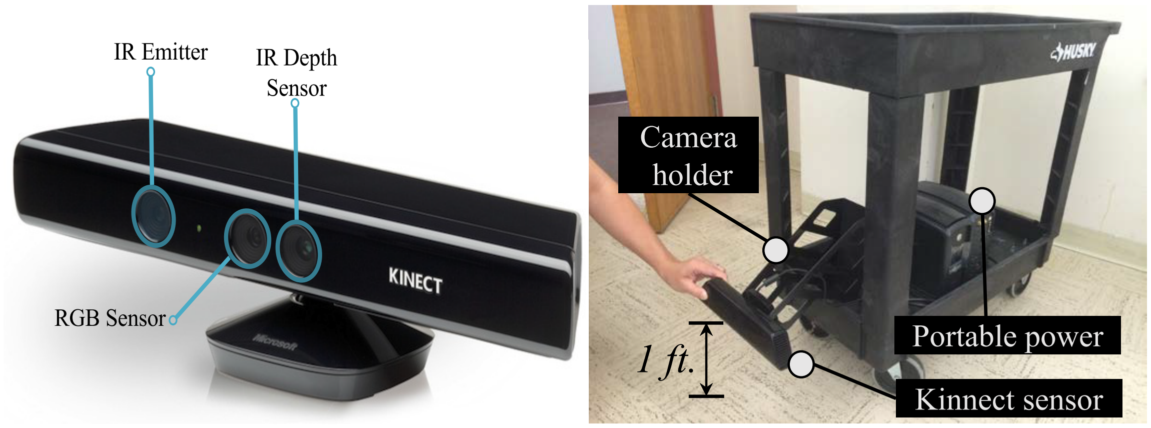


Figure 3.1: The first generation of Microsoft Kinect sensor (left) and the data collection setup (right).

Kinect as a navigation sensor for mobile robotics. Inspired by the great success of Kinect applications in these areas, this research proposed an innovative system for pavement surface reconstruction and cracking recognition.

A technical challenge of involving 3D sensors during pavement condition evaluations lies in the fact that these sensors inevitably induce more information to be processed. While there exist a wide range of algorithms for geometric analysis like manifold harmonics [VL08, LPG12, CLB+09] or spherical harmonics [WSCY16, KFR03], these methods are often *global* meaning they tend to obtain the most useful information based on the entire 3D model. This is clearly not the case of the analysis of pavement distress. We show that by only examining local sharp features, we can robustly and accurately extract key parameters associated with pavement cracking.

3.3 Kinect-based Data Collection

The Microsoft Kinect device (the first generation) is employed as the major sensor for data collection. Kinect was originally designed as a device for home entertainment since it enables human-computer interaction without additional controllers [SLPR13].

The Kinect sensor consists of an infrared (IR) laser emitter, an IR camera, and a regular RGB color camera as shown in Fig 3.1. Besides the traditional RGB sensing with the resolution of 640×480 pixels at 30 frames per second, Kinect is also capable of sensing the depth information by tracking the emitted IR rays. The geometry of the pavement surface can be further represented by converting the *level-set* surface representation [MSV95] into a triangle mesh, consisting of small inter-connected triangle faces using the marching cube algorithm [LC87].

In this study, the data of pavement cracks on road surfaces were collected at the University of New Mexico main campus and representative local streets and highways, including the segment of `Central Ave.` from `Washington St. NE` to `Broadway Blvd. SE` (a 23.1-mile long multi-lane highway, both Eastbound and Westbound directions), the segment of `Lomas Blvd. NE` from `San Mateo Blvd. NE` to `University Blvd. NE` (a 14.2-mile long multi-lane highway, both Eastbound and Westbound directions), the segment of `Girard Blvd. SE` from `Indian School Rd. NE` to `Gibson Blvd. SE` (a 22.5-mile long two-lane highways, both Northbound and Southbound directions) and the segment of `Yale Blvd. SE` from `Central Ave. SE` to `Gibson Blvd. SE` (a 3.5-mile long two-lane highway, both Northbound and Southbound directions) in the City of Albuquerque, NM. To facilitate the procedure of data collection, a mobile data collection stand was built for mounting the Kinect during the pavement data collection on-site as shown in Fig. 3.1. The camera holder fits the base of the attached Kinect sensor and gets close to the floor, which allows us to use the *near mode* of the Kinect fusion [IKH+11] and improves the result. Two portable power supplies are also equipped on the stand. A `Lenovo Thinkpad T430` laptop computer equipped with an `Intel i7 CPU` and `16G RAM` was connected to the Kinect. Note that slight oscillations of the Kinect sensor during the data collection do not affect the accuracy or quality of the final reconstruction as the camera's position and orientation can be dynamically tracked during the Kinect fusion [IKH+11]. Due to the hardware limitation, excessive darkness or brightness in the environmental ambient will degenerate the performance of the RGB camera. However, this issue can be easily fixed by adding artificial illumination when boarded on a moving van.

Three types of pavement cracks were measured: 1) Longitudinal cracking refers to cracks that are predominantly parallel to the pavement centerline (or traffic direction) [BHM+12]; 2) Transverse cracking is the ones are predominantly perpendicular to the pavement centerline; 3) Alligator cracking corresponds to the cracks occur in areas subjected to repeated traffic loadings, especially along the wheel paths. In the early development stages, alligator cracking can appear as a series of interconnected seams. Eventually, they morph into many-sided, sharp-angled pieces, usually less than one foot on the longest side, characterized by a chicken wire/alligator skin pattern in the later stages. For each type of crack, 339 to 385 sample data were collected for cracking detection and analysis with a total of crack samples. The amount of cracking samples for each type on each severity was determined based on previous studies and statistical experiences [JJMBG12, VESS13]. Each record of a single pavement crack sample consists of a 3D mesh based on the captured Kinect depth streams. An evaluation from human experts was conducted as the ground truth through dedicated camera photographs taken for the same samples (Fig. 3.2). In order to distinguish longitudinal cracks from the transverse ones, the axis of a depth frame was aligned with the traffic direction. Thus, the direction of longitudinal cracks in the depth frame captured by Kinect is vertical and that of transverse cracks is horizontal. All the pavement data were collected on the dry surface of asphalt concrete pavement.

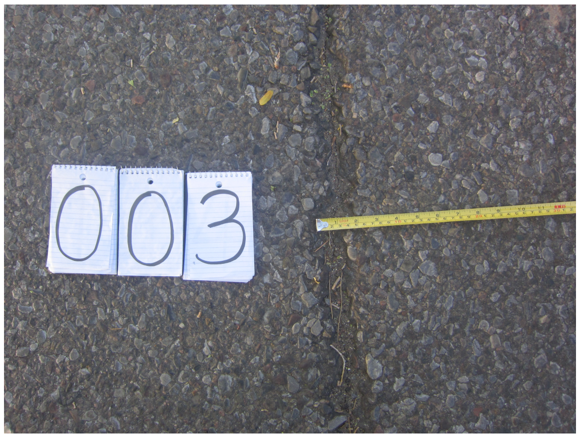


Figure 3.2: The expert evaluation is based on the on-site photograph.

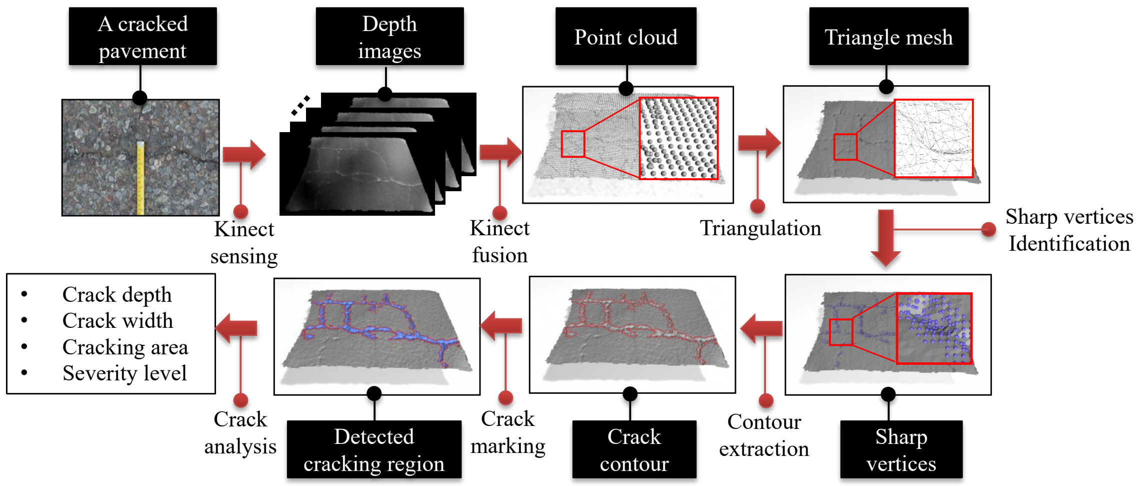


Figure 3.3: An overview of the proposal framework of crack analysis.

3.4 Research Methodology

3.4.1 An Overview of System Framework

This paper aims to utilize the Microsoft Kinect to reconstruct pavement surfaces and capture geometric information of pavement cracking, including crack width, length, and depth. As sketched in Fig. 3.3, we developed a series of algorithms to facilitate an automatic identification of distress severities of three major types of pavement cracks to provide necessary information for pavement condition evaluation. Observing the fact that pavement cracks inevitably undermine the smoothness of the surface geometry, we devised a *local* algorithm that automatically screens all the potential *sharp vertices* on the mesh, where a salient surface geometry variation exists. This is accomplished by analyzing the distribution of normals of a small neighboring region surrounding a mesh vertex being examined. A breadth-first search (BFS) is used to obtain connected components out of all the sharp vertices. The cracking region can then be identified as the covered area of the largest connected vertices. The geometric parameters, such as the width, length, and depth of the cracks, are then calculated. Each step in Fig. 3.3 will be detailed in the following sub-sections.

3.4.2 Depth Retrieval and Surface Reconstruction

Although the Kinect sensor provides a fast dense depth sampling of its target's surface, the raw data could contain a significant amount of high-frequency noises and missing captures (e.g. holes on the surface). In addition, one Kinect frame is only able to sense the depth information of a small region, which is far from sufficient to completely cover an entire cracking site. To resolve this problem, we used a technique named Kinect fusion [JJMBG12]. Kinect fusion "merges" the depth information from multiple Kinect frames. Specifically, for an incoming depth frame from Kinect sensor Kinect fusion first computes the corresponding camera pose, which can be encoded as a 4 by 4 homogeneous matrix. After unprojecting the depth points into 3D, a multi-resolution iterative closest point (ICP) method [BM92] is used to align the frame into a global 3D model (which is the pavement surface in our system) represented using the volumetric truncated signed distance function (TSDF) [CL96]. The constructed pavement mesh contains 3D geometry information of the pavement surface (we refer readers to related studies [NIH+11, JJMBG12] for more detailed explanations of algorithmic procedures of Kinect fusion).

3.4.3 Feature Extraction

The concept of k-ring neighbors of a vertex v on the mesh is frequently utilized in our feature extraction stage. Let $\mathscr{M}(\mathscr{E}, \mathscr{V})$ denote the triangle mesh, where $\mathscr{E}$ and $\mathscr{V}$ are the sets of edges and vertices on $\mathscr{M}$. The *one-ring neighbor* $\mathscr{N}_1^v$ of vertex $v \in \mathscr{V}$ is the set of vertices such that $\forall v_i \in \mathscr{N}_1^v$, $\langle v, v_i \rangle \in \mathscr{E}$. The one-ring neighbor of all the vertices on the mesh can be easily found by iterating all the triangles on the

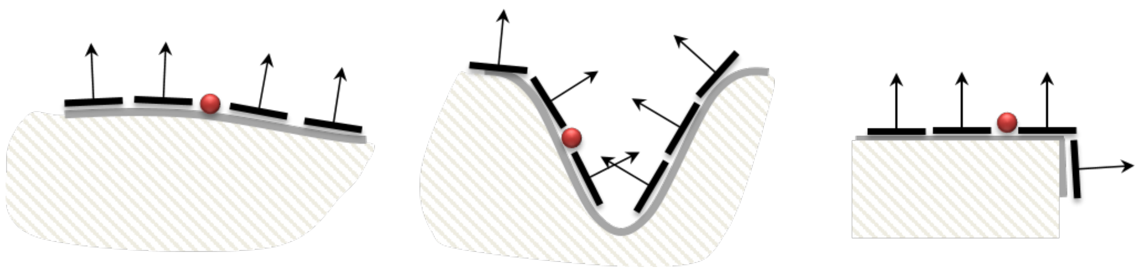


Figure 3.4: The normal directions close to a vertex on a smooth pavement surface are mostly in parallel (left). When it is close to a crack, normals are irregularly scattered (middle). A non-uniform normal distribution does not necessarily mean a sharp vertex (right).

mesh once, which is clearly an $\mathbf{O}(N)$ operation. Similarly, the k-ring neighbor $\mathscr{N}_k^v$ of vertex v is defined as the set of vertices such that they can be reached by traveling from v through at most k edges. The k-ring face neighbor $\mathscr{F}_k^v$ is the set of triangles such that each of them holds at least one element in $\mathscr{N}_k^v$. As such neighboring information of vertices will be used repeatedly, we assign each vertex a linked list storing $\mathscr{N}_k^v$ and $\mathscr{F}_k^v$ immediately after the triangle mesh is reconstructed.

The next step is to identify the locations corresponding to the cracks on the constructed 3D surface. It is assumed that an intact pavement surface is smooth indicating that the normal of nearby triangles should be in the similar direction. On the other hand, the location close to the crack is more likely to have irregular distribution of normals as shown in Fig. 3.4. To evaluate the smoothness of the neighbor region of vertex v, a flatness test is performed by computing the area-weighted average normal $\bar{\mathbf{n}}$ for all the triangles that are close to v, e.g. using the k-ring face neighbor, such that:

$$\bar{\mathbf{n}} = [\sum_{F_i \in \mathscr{F}_k^v} A(F_i) \cdot \mathbf{n}(F_i)] / [\sum_{F_i \in \mathscr{F}_k^v} A(F_i)] \tag{3.1}$$

Here, $\mathbf{n}(F_i) \in \mathbb{R}^3$ and $A(F_i)$ denote the unit normal vector and the area of the triangle $F_i \in \mathscr{F}_k^v$ respectively. Afterwards, we computed the "distance" between $\mathbf{n}(F_i)$ and

$\bar{\mathbf{n}}$ using the angle α_i between them:

$$\alpha_i = \arccos\left[\bar{\mathbf{n}} \cdot \mathbf{n}(F_i)/|\bar{\mathbf{n}}|\right] \tag{3.2}$$

Finally, the standard deviations (SD) among all the calculated α_i are evaluated. If the SD is lower than a given threshold, the region determined by $\mathscr{F}_k^v$ is regarded as *flat* and being free of cracks. Otherwise, F_i could potentially be associated with sharp features (e.g. edges of the crack) and further analysis would be necessary. We found that the flatness test based on two or three-ring neighbors can effectively remove most smooth vertices.

The discrete Gauss map (DGM) [DMSB00] is employed for further investigations of the local geometry feature associated with v that fails the flatness test. In DGM, a unit sphere centered at v is defined and each $F_i \in \mathscr{F}_k^v$ is mapped to a point p_i on the sphere's surface by travelling from v along $\mathbf{n}(F_i)$ for a unit distance. This can be computed via:

$$\mathbf{p}_i \triangleq DGM(F_i, v) = \mathbf{v} + \mathbf{n}(F_i), \quad F_i \in \mathscr{F}_k^v, \tag{3.3}$$

where $\mathbf{p}_i, \mathbf{v} \in \mathbb{R}^3$ are the 3D positions of p_i and v. Fig. 3.5 shows an illustrative example of the DGM for a vertex on the mesh. We reorganize the mapped points on the sphere into clusters such that points within a cluster are closer to each other. It is clear that if v happens to sit on the intersection of two planes, its neighboring faces should hold two distinctive normal directions. Accordingly, its DGM points can be grouped into two clusters. Similarly, three DGM clusters indicate an intersection by three planes of different orientations. However, if the number

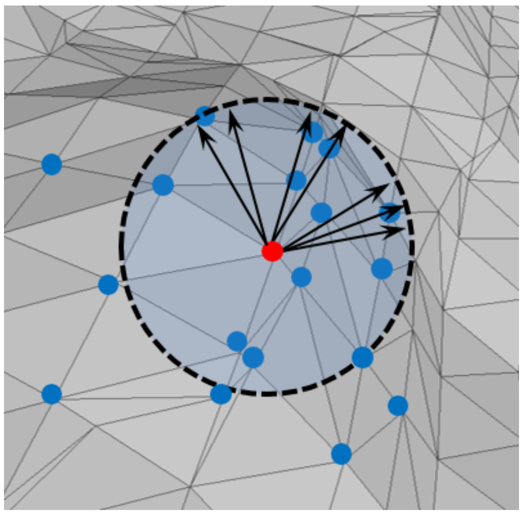


Figure 3.5: DGM of a vertex.

of resulting clusters is larger than four, it is more likely that is on a rough surface rather than a sharp feature as we rarely have intersections of more than four planes on the roadway pavement. The major calculation in the DGM clustering is to measure the "distance" between two DGM points. As each point actually represents a normal vector of a triangle in $\mathscr{F}_k^v$, the regular Euclidean distance is obviously not a good choice. Alternatively, we use the *geodesic distance* on the sphere (e.g. the great-arc length), which equals the minimal angle between the two normal vectors and can be computed as:

$$dist_g\left(p_i, p_j\right) = \arccos\left(\mathbf{n}(F_i) \cdot \mathbf{n}(F_j)\right). \tag{3.4}$$

At the beginning, each DGM point is assigned to a different cluster. The distance between two clusters $\mathscr{C}_i$ and $\mathscr{C}_j$ is defined as the maximum distance between all the DGM point pairs from each cluster:

$$dist\left(\mathscr{C}_i, \mathscr{C}_j\right) = \max_{p_n \in \mathscr{C}_i, p_m \in \mathscr{C}_j} dist_g\left(p_n, p_m\right), \tag{3.5}$$

where p_n and p_m are DGM points from $\mathscr{C}_i$ and $\mathscr{C}_j$ respectively. As long as $dist(\mathscr{C}_i, \mathscr{C}_j)$ is smaller than a sensitivity parameter α_t, they will be merged into a new cluster. We keep merging the clusters until the distance between any pair of the clusters is larger than α_t. If the final number of the clusters is between two and four, v is considered a *sharp vertex.*

3.4.4 Crack Analysis

Although our method is able to mark most vertices on cracks as sharp vertices, some vertices away from cracks may also be mistakenly labeled due to the regional pavement roughness. Therefore, we need to further extract vertices that truly belong to the crack. This was achieved by applying a BFS over all the sharp vertices detected in the previous step based on the assumption that the crack region should

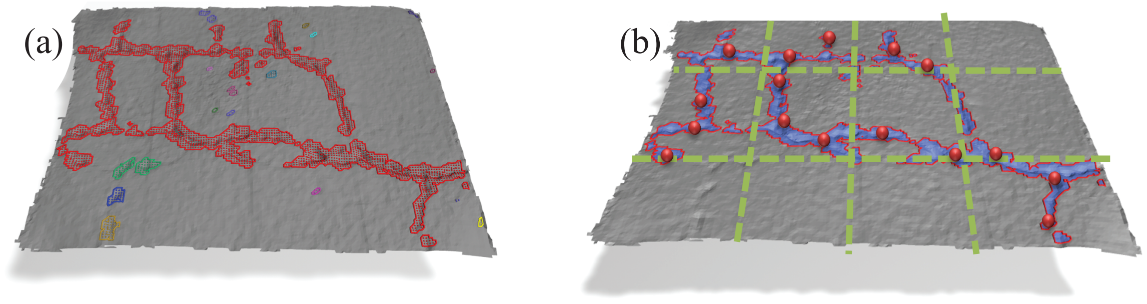


Figure 3.6: (a) The result of BFS on a pavement mesh. The red regions are the large connected component. Other small components in other colors are discarded. (b) Crack depth evaluation by grids. At each grid, the deepest sharp vertices are picked (red spheres) for local depth evaluation.

be the most dominant geometry feature on the pavement segment of interest. BFS algorithm retrieves all the connected components on the mesh, where a connected component is a subset of vertices and edges such that any two vertices can be reached through its edge set. The connected components of small size (e.g. smaller than $1,000$ sharp vertices) are most likely associated with some minor surface dents rather than cracks. Therefore, they are discarded (as shown in Fig. 3.6 (a)). The triangle incident to a sharp vertex is labeled as *sharp face*. Due to the strong connectivity of the connected component, it is guaranteed that all the triangles associated to a connected component are also connected and formed a sub-mesh or the cracking region. The contour of the crack region can be easily extracted by iterating all the edges: a contour edge is the one shared by two triangles such that one of the triangles is a sharp feature face while the other one is not. In order to make the framework directly useful for the pavement evaluation, detailed parameters and statistics, such as the width, length, and depth of cracks, must be automatically reported out of the crack region marked. This feature is also supported in our framework.

Crack Depth

It seems that the crack depth could be directly obtained by looking at the values of the sharp vertices within a crack region. Unfortunately, it is not always the case that the pavement surface perfectly aligns with the plane of the camera coordinate frame (CCF). In most situations, the pavement of the road formed an arch and the depth of the crack was actually the distance from the valley of the crack to the tangent plane of the surface. Based on this observation, we used the least square fitting (LSF) sphere surface to approximate the pavement arch. The LSF sphere is described by the standard sphere equation:

$$x^2 + y^2 + z^2 - Ax - By - Cz + D = 0,$$

where A, B, C, D are four unknown coefficients to be determined. The quadratic equation is optimal (best fitting) when the sum of the squared distance from vertices to sphere surface is minimized:

$$\arg \min_{A,B,C,D} E, \quad E = \sum \left(x_i^2 + y_i^2 + z_i^2 - Ax_i - By_i - Cz_i + D\right)^2,$$

where x_i, y_i and z_i are the x, y and z coordinates of a vertex. Because sharp vertices are located at the valley/edge of the crack far away from the pavement surface, they should not participate in LSF sphere equation evaluation. Accordingly, the summation E only takes over all the non-sharp vertices. The unknowns A, B, C and D can be solved by setting the gradient of E as $\mathbf{0}$:

$$\Delta E = \left[\frac{\partial E}{\partial A}, \frac{\partial E}{\partial B}, \frac{\partial E}{\partial C}, \frac{\partial E}{\partial D}\right] = \mathbf{0},$$

which yields a 4×4 linear system:

$$\begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i z_i & -\sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i z_i & -\sum y_i \\ \sum x_i z_i & \sum y_i z_i & \sum z_i^2 & -\sum z_i \\ -\sum x_i & -\sum y_i & -\sum z_i & \sum 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} \sum x_i(x_i^2 + y_i^2 + z_i^2) \\ \sum y_i(x_i^2 + y_i^2 + z_i^2) \\ \sum z_i(x_i^2 + y_i^2 + z_i^2) \\ -\sum x_i(x_i^2 + y_i^2 + z_i^2) \end{bmatrix} \tag{3.6}$$

Lastly, the depth of a vertex v on the mesh is defined as the distance to the LSF sphere surface:

$$depth(v) = R - \sqrt{(x - x_o)^2 + (y - y_o)^2 + (z - z_o)^2}, \tag{3.7}$$

where x_o, y_o and z_o are the coordinates of the sphere center and R is the sphere radius. The crack depths at different regions could be different. We regularly partitioned the pavement mesh by grids. The top 5% of the deepest sharp vertices computed via Eq. (3.7) serve as representative depth samples at each local grid cell. The average of them is then used for the final crack depth estimation as shown in Fig. 3.6 (b).

Crack Width

Evaluating the average crack width is challenging especially for cracks with irregular patterns. As shown in Fig. 3.7, the local deepest sharp vertex (the red dot) within a grid cell is assumed to be located at the valley of the crack. The projection of the vector pointing from the sharp vertex to its closest surface vertex on the LSF sphere surface provides us a reasonable approximation of the half width of the crack. As a result, the local crack width at the grid was computed as:

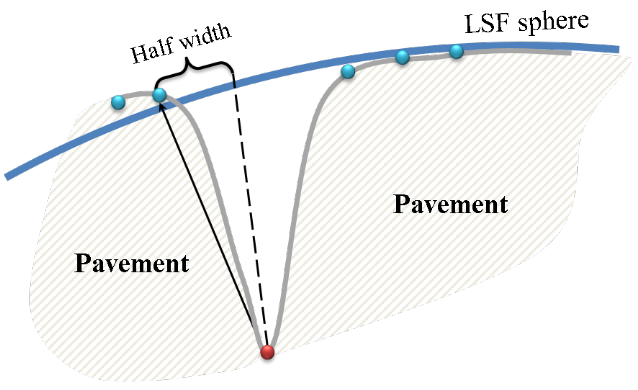


Figure 3.7: Width measurements.

$$width(v) = 2|\mathbf{v}' - \mathbf{v}'_S|, \tag{3.8}$$

where $\mathbf{v}', \mathbf{v}'_S \in \mathbb{R}^3$ are 3D positions of the deepest sharp vertex and its closest projected crack boundary vertex. In our implementation, we use the three nearest crack boundary vertices for a better width approximation.

Crack Length & Area

The area of the crack is just the summation of the area of all the sharp faces projected to the LSF sphere. The length can be computed by dividing the crack area by its average width. Finally, the severity of the crack can be estimated based on the evaluated crack parameters.

3.5 Experimental Tests and Discussions

3.5.1 Surface Reconstruction

With the help of Kinect fusion technique, the 3D reconstruction can be made for a wide pavement surface area. Indeed, we can reconstruct arbitrarily wide and lengthy pavement as long as there is sufficient hard drive space. After the mesh reconstruction is completed, the aforementioned feature detection and crack analysis algorithm will be applied. It is also easy to see that, all the calculation for extracting crack's geometry is essentially local, meaning the entire analysis is an $\mathbf{O}(N)$ linear algorithm. The calculated crack parameters (width, length and depth) are used following the existing flexible pavement evaluation standard in New Mexico [BHM+12] to assess the severity of each crack sample. Such results are further compared with manual severity estimation by experts for algorithm performance assessment.

We also created an online database using a `WebGL` based interface (Fig. 3.8), which can be assessed by the readers for free to further test our algorithms. The red spots on the map interface (supported by `Google Map API`) on the left corresponds to a crack site and its street view is provided at the bottom left corner. In the middle, the list of data from this site is provided and users can click on any of them to download it. Each item consists of a 3D mesh as shown in the rendering panel on the right and a photo with sample ID.

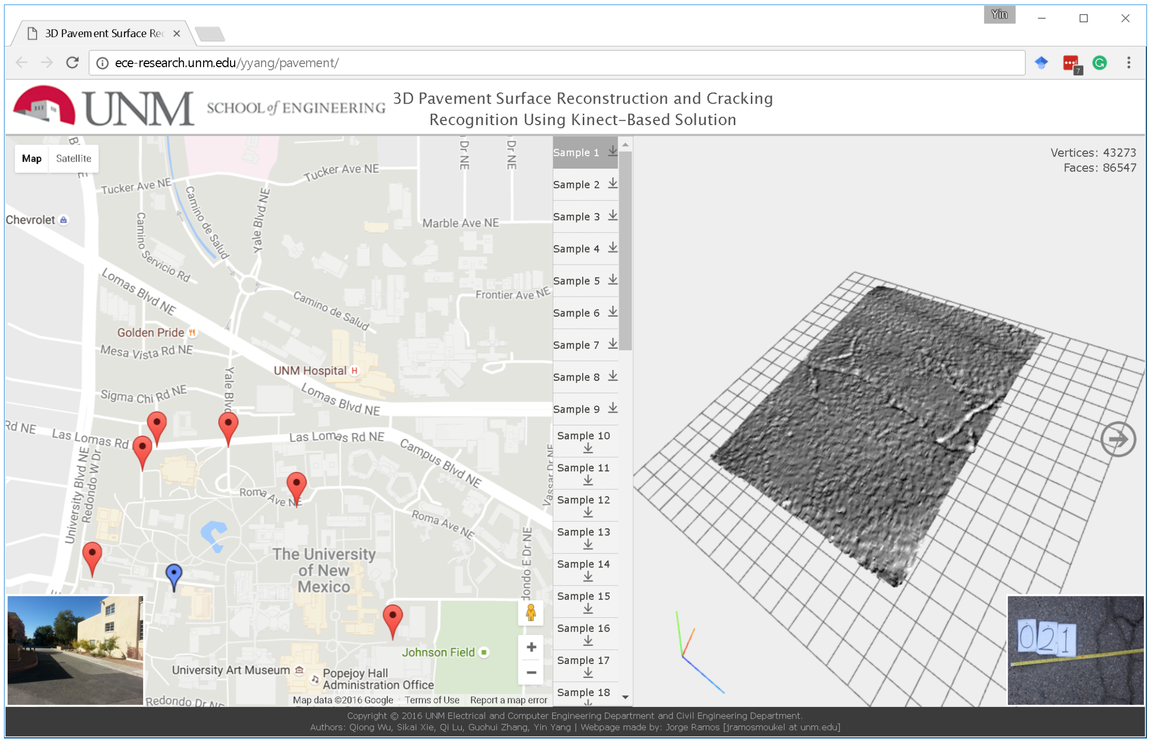

3D Pavement Surface Re
ece-research.unm.edu/yyang/pavement/
UNM SCHOOL of ENGINEERING
3D Pavement Surface Reconstruction and Cracking Recognition Using Kinect–Based Solution
Map
Satellite
Sample 1
Sample 2
Sample 3
Sample 4
Sample 5
Sample 6
Sample 7
Sample 8
Sample 9
Sample 10
Sample 11
Sample 12
Sample 13
Sample 14
Sample 15
Sample 16
Sample 17
Sample 18
Vertices: 43273
Faces: 86547
The University of New Mexico
UNM Hospital
Johnson Field
University Art Museum
Popejoy Hall
Administration Office
Map data ©2016 Google Terms of Use Report a map error
Copyright © 2016 UNM Electrical and Computer Engineering Department and Civil Engineering Department.
Authors: Qiong Wu, Sikai Xie, Qi Lu, Guohui Zhang, Yin Yang | Webpage made by: Jorge Ramos [jramosmoukel at unm.edu]

Figure 3.8: A WebGL based database for collected cracking samples, available at http://ece-research.unm.edu/yyang/pavement/.

3.5.2 Accuracy Experiment

The suggested range of the first generation Kinect sensor is between 0.3 to 3 meters (i.e. 11.9–118 inches) when the *near mode* is on. The phantom study tests the accuracy of the Kinect senor with a simple man-made cracking surface. As shown in the Fig. 3.9, the phantom was made of a cardboard with a one-inch wide and one-inch deep artificial dent at its middle. We compare the width/depth information calculated with the proposed algorithm on the digital reconstruction from the Kinect. Since the ground truth value is precisely known, this simple phantom study allows a quantitative understanding of the accuracy of the proposed algorithm. The result is reported in Fig. 3.9, where we plot the relation between the Kinect-phantom distance and the relative error between the calculated depth/width and the ground truth. It can be seen from the figure that the relative error of our algorithm based on the Kinect

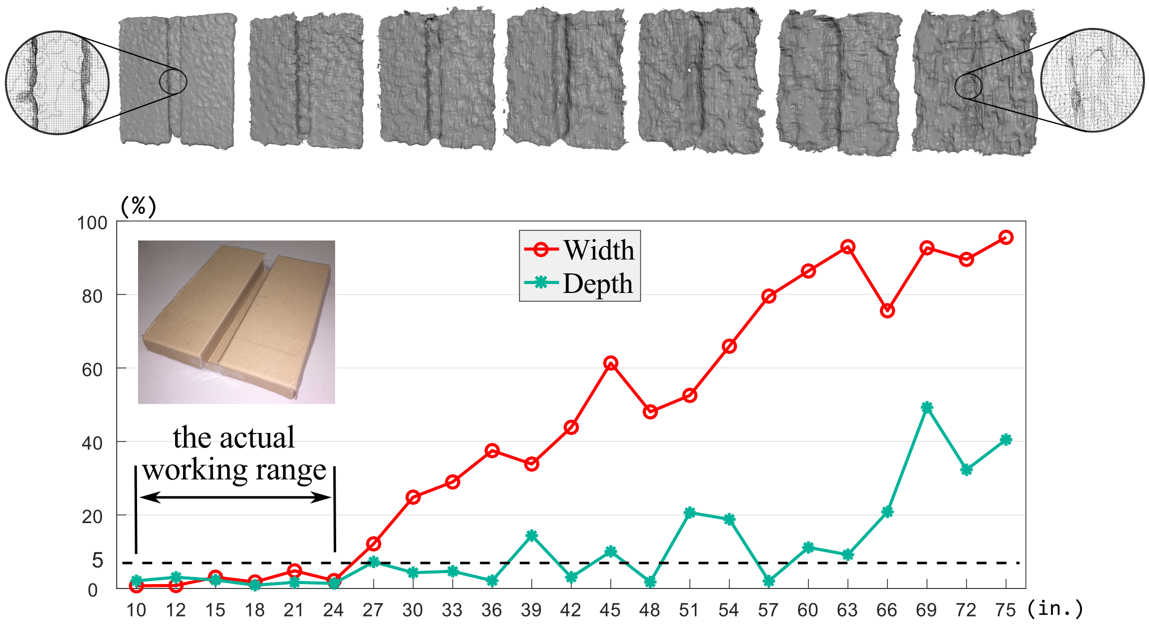


Figure 3.9: We test the accuracy of the proposed algorithm using a standard man-made cracking surface. The accuracy is typically below 5% if the Kinect-phantom distance is between 10 to 30 inches. The reconstructed surfaces are given in top.

fusion is always less than 5% if the Kinect-phantom distance is between 10 inches to 30 inches, which is the typical working distance in our data collection. It is not surprising that if this distance increases, the quality of the resulting reconstruction is downgraded (as shown in Fig. 3.9 top). Thus the relative error goes up accordingly.

To further illustrate the versatility of our algorithm, we also did a side-by-side comparison applying the proposed method to 3D surface geometries obtained from both LIDAR and Kinect. For the LIDAR, a Delaunay triangulation [LS80] was performed the construct the corresponding mesh. The result shows that the quality of Kinect data is not as superior as the ones from the professor LIDAR sensor. However, the cracking region is still correctly detected regardless the resolutions of the input meshes. The experiment validates the robustness of our algorithm.

3.5.3 Test Results for Various Cracking Detection

The proposed system is able to capture the shape information of various cracks. Before being analyzed by the proposed Kinect fusion and crack detection technique, all the collected crack sample data were manually examined by trained pavement inspectors, and their severity evaluation results, after agreeable adjustments, were used as observed truth data for performance assessment purposes. The data collection was carried by three Ph.D. students and two professors. All the members went through a two-day training workshop provided by NMDOT. The geometry measure of the cracks were all obtained in the field. Table 3.1 reports the performance of the proposed approach and the confusion matrix for transverse cracking regarding each severity as well as failure detection. Tables 3.2 and 3.3 demonstrate these summaries for longitudinal cracking and alligator cracking. In these tables, each row represents the number of observed instances for each severity level, and each column illustrates the number of predicted instance for each severity and failure detection. The last two columns list the *true positive rate* (TPR) and *failure detection rate* (FDR) for each crack severity. TPR measures the proportion of actual positives which are correctly identified as such (i.e. the proportion of Severity 1 samples that are correctly identified as Severity 1), and FDR denotes the proportion of records that failed to identify. The three tables show that the proposed method was able to correctly identify 78.27% of longitudinal cracking, which is the highest of all the three major cracking types, and it performs relatively inferiorly on alligator cracking detection, with the lowest TPR of 55.16%.

In the meantime, the proposed method failed to identify a small amount of collected samples for each cracking type, indicated by the overall FDR. With comparable sample size for each cracking type, the proposed approach has the highest amount (42 samples) and FDR (10.91%) for transverse cracking, and close recognition performance on longitudinal cracking (24 samples, 6.69%) and alligator cracking (30

Table 3.1: Transverse cracking severity detection results.

Observed Samples by Severity	**Predicted Samples by Severity**					
	Severity 1 (104)	Severity 2 (149)	Severity 3 (90)	**FD** (42)	**TPR**	**FDR**
Severity 1 (160)	101	45	1	13	63.13%	8.13%
Severity 2 (113)	3	81	11	18	71.68%	15.93%
Severity 3 (112)	0	23	78	11	69.64%	9.82%
Total 385	260 (Total correct predictions)			42	67.53%	10.91%

Table 3.2: Longitudinal cracking severity detection results.

Observed Samples by Severity	**Predicted Samples by Severity**					
	Severity 1 (110)	Severity 2 (146)	Severity 3 (79)	**FD** (42)	**TPR**	**FDR**
Severity 1 (144)	109	31	1	3	75.69%	2.08%
Severity 2 (113)	1	99	5	8	87.61%	7.08%
Severity 3 (102)	0	16	73	13	71.57%	12.75%
Total 359	281 (Total correct predictions)			24	78.27%	6.69%

Table 3.3: Alligator cracking severity detection results.

Observed Samples by Severity	**Predicted Samples by Severity**					
	Severity 1 (69)	Severity 2 (142)	Severity 3 (98)	**FD** (30)	**TPR**	**FDR**
Severity 1 (119)	65	31	5	18	54.62%	15.13%
Severity 2 (128)	4	73	44	7	57.03%	5.47%
Severity 3 (92)	0	38	49	5	53.26%	5.43%
Total 339	187 (Total correct predictions)			30	55.16%	8.85%

samples, 8.85%). These results suggest that TPR and FDR are effective indices measuring the performance from certain aspects and should both be utilized for pavement cracking detection and evaluation.

The performance of the proposed method also varies for the same type of cracking with different severities. Taking Table 3.1 as an example, it shows that for transverse cracking, the proposed approach performs best on Severity 2 and is able to correctly classify 71.68% of all Severity 2 samples, followed by a comparable performance on Severity 3 (69.64%). It per-forms worst on Severity 1, with a TPR of 63.13%. This implies that the proposed approach is relatively better able to classify transverse cracks of higher severities. However, similar to the overall performance, it was also found that the pro-posed method is unable to recognize some of the cracks regarding each severity. Overall, the proposed approach is able to identify 89.09% (FDR=10.91%) of all the transverse crack samples and is capable of correctly classifying 67.53% of all the transverse cracks, indicating an acceptable prediction performance. As is shown in Table 3.1, this method fails to identify 8.13% of the Severity 1 transverse cracking, which is the least of all the three severities, followed by failure detections on Severity 3 (9.82%) and Severity 2 (15.93%). It is suggested that the proposed approach performs worst on recognizing Severity 2 transverse cracking, but in the meantime works best after Severity 2 cracking is recognized, as is shown by the TRP. The crack detection results for longitudinal cracking and alligator cracking could be interpreted analogously, and therefore are omitted in this discussion.

It is also displayed in Table 3.1 that the proposed method tends to overestimate transverse cracks in Severity 2 and underestimate those in Severities 1 and 3. Specifically, there are a considerable amount of misclassified instances in each pair of severities: 45 samples of Severity 1 are misclassified as Severity 2 and 1 sample of Severity 1 is misclassified as Severity 3; 3 Severity 2 records are misclassified as Severity 1 and 11 Severity 2 samples are misclassified as Severity 3; 23 Severity 3 records are misclassified as Severity 2. It is also revealed that there is overall only 1 misclassification between Severity 1 and Severity 3, indicating a significant discrepancy between these two severity levels and that the proposed method is effective in detecting this discrepancy. The misclassifications for longitudinal cracking and alligator cracking, which are illustrated in Table 3.2 and Table 3.3 respectively, could be analyzed accordingly.

According to existing flexible pavement evaluation protocol in New Mexico [BHM$^+$12], crack width is the major parameter used to define crack severity. Therefore, in this study, the crack width information is used in the severity classification process. However, it should be noted that both crack length and width are important parameters for pavement distress evaluation and therefore are also calculated by the proposed algorithm. Based on existing flexible pavement evaluation protocol, crack length is an important measurement to define pavement cracking extent. The cracking extent is not evaluated in this research, but the length information for each crack sample is extracted to verify the applicability of the proposed approach. Besides, the most critical problem affecting the pavement service life is the formation and growth of tracks due to physical stress and chemical deterioration. Therefore, crack depth is generally used a factor to determine pavement surface maintenance and rehabilitation [LMG02]. Taking this into account, this study also extracts crack depth information, which may provide instructive reference for pavement surface maintenance schedule optimization.

Fig. 3.10 shows the result of three successful detections of alligator cracking, longitudinal cracking, and transverse cracking, respectively. We can see that the crack region is accurately identified and cracking features are explicitly characterized. It is demonstrated that the Kinect fusion algorithm and crack detection techniques are able to capture the shape information of various cracks. However, our method could also underperform in some extreme cases, where the crack is shallow and some regular surface roughness could present as significant geometry variations as the crack does. Therefore, high-level noise (fake sharp vertices) could be observed in the failure example as the bottom row in Fig. 3.10. The crack has less depth and width magnitude compared to the successful example, indicating that crack depth and width are significant factors related to successful crack detection. It also indicates that our system performs better on higher severities than lower severities, which is reasonable as the high severity crack is usually associated with width, length or

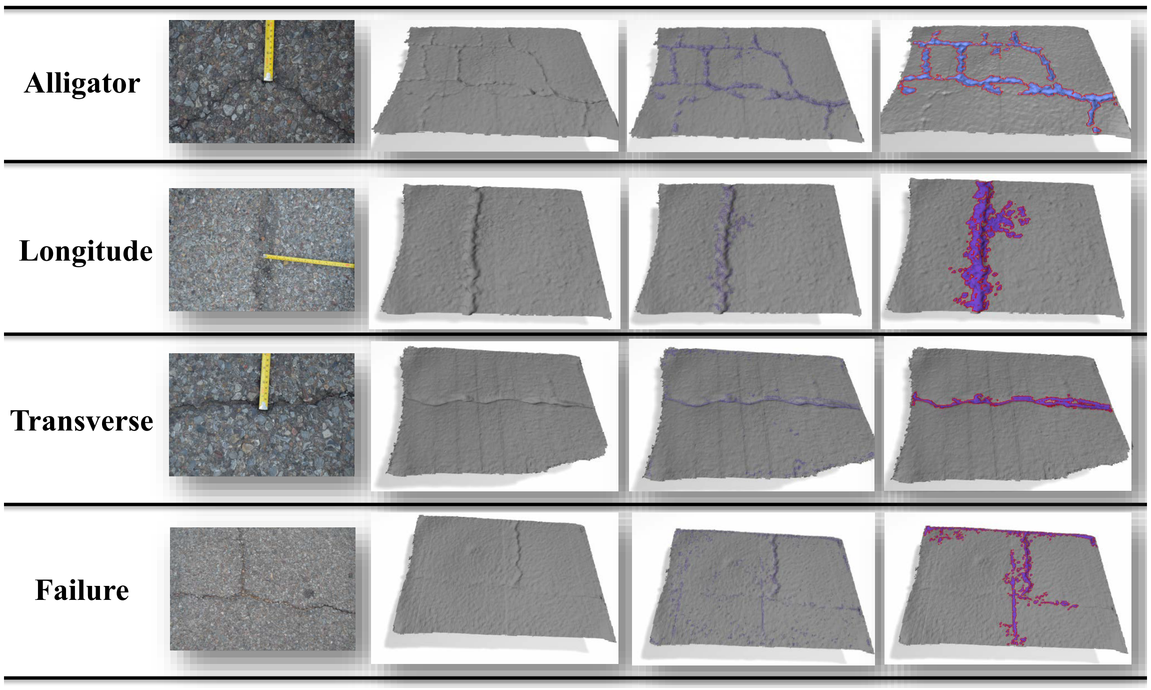


Figure 3.10: More results from our method.

depth of larger magnitude. Overall, the developed Kinect fusion technique and crack detection algorithms are able to detect three major types of pavement cracks. The proposed approach provides a viable alternative for pavement crack detection.

3.5.4 Limitations

While our experiment reports promising results, there still exist some limitations in the current version of our system, which leave us many exciting further directions to explore. First of all, we were using the first generation of Microsoft Kinect in this work. By the time of this paper submission, the second generation of Kinect had just been released, with higher depth resolution and frame rates. We will adapt our system to the latest Kinect hardware in the near future and a much more detailed surface reconstruction is expected. Second, our feature detection method works well for local sharp geometry features. However, lower performance was observed for subtle surface roughness and global shape variation (e.g. on a wide and smoothly curved surface). It is necessary to investigate new geometric analysis method to detect the cracks on the re-constructed pavement surface. Using spectral geometry analysis [HH09] is a promising further direction. Currently, we perform the crack analysis purely based on the reconstructed 3D mesh surface. We believe that by combining information from other data resources e.g. the classic RGB video camera [BVLH06], the accuracy of the pavement analysis can be further improved. We will also explore further possible enhancement of our current algorithm, for instance, to use extended ICP algorithm [HTB03] to improve the precision of the 3D construction from depth frames. Of course, the algorithm becomes more expensive and we may need to utilize a parallelization on general-purpose graphics processing unit (GPGPU) to further accelerate the processing [LGG01]. Moreover, there is still a limitation regarding Kinect field of view in this study, due to which the examination of crack extent is not applicable. A possible solution, as was proposed in a previous study, is to use an array of Kinect to cover larger areas in both longitudinal and transverse directions. We also plan to install the Kinect to the vehicle to collect more 3D geometry data of the local streets and highways. In order to do that, we need to study how to de-blur the depth data when the Kinect sensor undergoes a fast movement. This may be achieved by fusing information from multiple Kinect devices.

3.6 Conclusions

This study applies Microsoft Kinect, a consumer-level motion sensing input device to detect the geometric features, including width, length, and depth of different types of pavement distress. Research results indicate that Microsoft Kinect produces reliable geometric information of pavement distress and is able to report distress severity with a promising accuracy. Crack depth and width are significant factors related to successful crack detection indicated by the comparison of successful detection and failure detection examples, which demonstrates that Kinect crack detection algorithms perform better on higher severities than lower severities. Research limitations regarding the hardware constraint, universal application extensions, and data accessibility are also discussed. The main advantage of the proposed method includes two aspects: first, compared with the existing automatic pavement evaluation method, the proposed approach captures 3D pavement crack image and extract crack depth information, which is an important measurement to define pavement cracking severity. Besides, compared with recently proposed advanced 3D surface reconstruction techniques, the Kinect fusion technique is consumer-level efficient and practice-ready, and has shown great potential for mass implementation with further improvement. Due to the hardware constraints and complicated data processing it is unlikely that the Kinect could completely replace the current state-of-the-art systems for pavement condition evaluation. Nevertheless, we believe the proposed system still provides an applicable complementary solution for automatic pavement evaluation, pavement surface 3D reconstruction, and distress severity quantification.

Chapter 4

Computational Future Traffic Flow Simulation

4.1 Introduction

The concept of smart transportation has drawn more and more attention while addressing important challenges and concerns like traffic congestion, fuel consumption, air pollution and so on. Emerging Connected Vehicle (CV) and Autonomous Vehicle (AV) technologies can improve network-wide traffic safety, mobility, and operation efficiency through real-time Dedicated Short Range Communications (DSRC) based Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications [JWZ+13, HPL15, Nob10]. The research team at the University of Toronto including Alberto Leon-Garcia, Hans-Arno Jacobsen, Baher Adbulhai, etc. conducted pioneering work to establish the Connected Vehicles and Smart Transportation (CVST) portal to share live integrated traffic information in CV environments [LGJA+, LG11, KLG11, LGK11]. Many other researchers in the university investigated driving challenges and opportunities in AV-enabled traffic systems [Tic15, Uof]. According to the Research and Innovative Technology Administration (RITA) of the U.S. Department of Transportation (USDOT), 81% of all vehicle-involved crashes can be avoided or significantly mitigated based on CV techniques annually. Meanwhile, AV is capable of sensing its environment and self-piloting based on navigation hardware such as cameras, radar, Lidar, laser rangefinders, and GPS. AVs can much more accurately judge distances and velocities, attentively monitor their surroundings, and react instantly to emergent situations. By combining CV and AV technologies seamlessly, it is believed that Connected Autonomous Vehicle or CAV enabled traffic systems can revolutionize the existing understanding of vehicle-infrastructure interactions and network-wide traffic system operations. However, the existing traffic theory becomes awkward when comes to the context of CAV. Existing traffic flow models [LW55, Ric56, Ros88, PBHS89, MBL84, Zha02, LPMS13, ADR16, WMMJ17] were developed for Human Driven Vehicle (HDV)-based traffic flow operations based on one-way coupled vehicle interactions adopted in classic car-following models (a following vehicle adjusts its operation conditions, such as acceleration/deceleration only based on its leading vehicle's position, relative speed difference, etc.). To incorporate the lateral traffic flow operations, additional lane-changing models must be involved. Enabled by CAVs, the two-way communication and collaborative linkages among CAVs can greatly facilitate us to formulate the mutually-coupled vehicle interactions (not only a following vehicle will be impacted by its leading vehicle, but also the leading vehicle will be impacted by its following vehicle and its surrounding vehicles too) for CAV-enabled traffic flow. This feature implies that CAVs are expected to move freely along both the longitudinal and lateral direction and a two-dimensional traffic flow model could be more reasonable. Currently, an aggregated macroscopic model for CAV-based traffic is still an under-investigated problem.

Motivated by this fact, we present a new traffic flow model for mutually-coupled vehicles enabled by CAV techniques. Under this scenario, an individual vehicle spontaneously seeks for its local optimal configurations (Generally speaking, local optimal configurations mean the best position and velocity that a CAV can have in order to, for example, reach the highest possible traveling speed or have the shortest traveling time. The objective is to improve the traffic mobility and maximize the transportation capacity.) based on the real-time information shared/obtained from the surroundings. Such behavior exaggerates intrinsic physics traits of the system, making the CAV-enabled traffic resemble other real-world continuum systems like fluid or molecular systems. We use several virtual internal and external forces to describe the two-way vehicle-vehicle interaction. By leveraging the Newton's second law of motion, our model naturally preserves the traffic volume, which is required as a macroscopic traffic flow model, and automatically handles both the longitudinal and lateral traffic operations. While our model is designed for CAV-enabled traffic, it can be easily rolled back to simulate the regular traffic of human drivers. The experiment shows that the proposed model describes the real-world traffic behavior well. Therefore, we consider the proposed model a generalization of the existing traffic theory. We also develop a Smoothed Particle Hydrodynamics or SPH based numerical simulation and an interactive traffic visualization framework.

The rest of this chapter is organized as follows: a brief literature review is given in Section 4.2. Section 4.3 explains our force-driven model and how it can be numerically simulated using SPH. The system implementation details are provided in Section 4.4, followed by the discussion of experimental results in Section 4.5. This research effort is concluded with future work in Section 4.6.

4.2 Related Work

CAV technology has been developed and demonstrated great potential to transform the way people travel through the interconnected network including cars, buses, trucks, trains, traffic signals, cell phones, and other devices [Tec13]. Litman concluded that AVs will consist of about 50% of vehicle fleet, 90% of vehicle sales, and 65% of all vehicles by 2050 [Lit14], although the true anticipated market share needs more solid verications and investigations as a research question. As an echo to such rapid deployment of CAV, many research efforts have been devoted to explore how CAV technology can improve the existing transportation system. Ni and colleagues proposed a car-following model incorporating the effects of CV technology and investigated the highway capacity gain [NLAW11]. Acceleration-based connected cruise control was proposed to increase roadway traffic mobility through wireless V2V communication [WOJK16, JO14]. An algorithm using CV data was proposed to minimize the vehicle delay and queue length at intersections [FHKZ15]. Multiple studies have been conducted on applications of CV technology, including traffic monitoring [DB08], ramp metering [Par08], route guidance [THK16, KTY16], traffic signal control [LP12, KIH$^+$15, ZZSM16, TZWZ15], vehicle-infrastructure-integration implementation issues [LTTA11], and public transportation [LLC16]. While most applications of CV technologies were assumed to be effective based on perfect V2V and V2I communications, it was pointed out that wireless communications could also experience packet delays/drops, which might lead to a serious downgrade of CV applications [DD15, OI15]. In CAV-enabled traffic systems, vehicle interactions are dramatically changed from one-way to all-directional neighborhood-wide information dissemination, which calls for a new macroscopic traffic flow formulation [MBSEF14].

Traditional continuum traffic flow models were pioneered by Lighthill and Whitham [LW55] and Richards [Ric56] to mathematically describe macroscopic traffic flow operations (known as the LWR model) based on a nonlinear conservation law. Many researchers introduced other high-order traffic flow models [MBL84, Ros88, PBHS89]. In recent work, Cheng and colleagues [CGW17] proposed a new continuum traffic flow model incorporating the effects of drivers' timid and aggressive behavior. Andreianov and Donadello [ADR16] employed point constraints in a second-order traffic flow model for modeling some traffic conditions, for instance, traffic signals.

Wang and colleagues [WMMJ17] used the energy conservation law to develop a two-regime speed-density formulation for the first-order traffic flow model. Zheng and colleagues [ZHH17] proposed a flexible traffic stream model connecting four driving behavior-related parameters to the fundamental diagram (i.e. the fundamental relationships among the flow, speed and density). All those macroscopic models are one-dimensional and focus on conventional traffic.

On the other hand, microscopic traffic flow models, or agent-based models, describe the car-following behavior and the lane-changing behavior of individual vehicles. There have been different microscopic models due to their specific assumptions of the flow behavior [GHP59]. The acceleration of the following vehicle is a function of a calibration constant, a reaction time, its velocity, and the velocity of the preceding vehicle relative to it. Recently, Asaithambi and colleagues [AKT16] provided a review of current microscopic driving behavior models and discussed both the longitudinal and lateral movements in mixed traffic. Guo and colleagues [GZY+17] proposed an improved car-following model that incorporates the effects of multiple preceding vehicles' velocity fluctuation feedback. Li and colleagues [LXXQ17] introduced a sensitivity factor to the classic car-following model so that the vehicles in traffic are divided into two types as low- and high-sensitivity vehicles and used the linear stability theory to analyze the stability of the new model. To find out the relationship between the microscopic and macroscopic traffic flow model, Garavello and Piccoli [GP17] coupled the car-following model to the LWR model through suitable boundary conditions, for example, at the location $x = 0$. Holden and Risebro [HR17] proved that the car-following model converges to the LWR model when traffic becomes dense. To calibrate the model parameters, Chiappone and colleagues [CGG+16] used a genetic algorithm based on specific fundamental diagrams. In their method, the calibration is formulated as an optimization problem whose objective is to minimize the difference between the simulated and real data. Given the trajectory data collected from CVs, Zhu and colleagues [ZU17] proposed an optimal estimation approach for calibrating the car-following behavior in a CVs and regular vehicles mixed traffic environment. Zhong and colleagues [ZFS$^+$16] adopted two approaches, the Cross-Entropy Method (CEM) and the Probabilistic Sensitivity Analysis (PSA) algorithm, for the calibration. The former is able to find the optimal set of the parameters, while the latter is applied to identify the most important parameters in order to reduce the computational burden. To predict the traffic flow, Zhao and Sun [ZS16] developed a fourth-order Gaussian process dynamical model (GPDM) which is a unsupervised learning method suitable for modeling dynamic real-world traffic data. Their experiments showed that the proposed method outperformed the existing methods for traffic flow predictions.

The existing traffic simulation packages VISSIM, AIMSUN, Paramics, etc. are discrete event-driven, agent-based simulation applications to mimic how the traffic system operates today. Many other outstanding simulation model applications include [ZMW14,LZK$^+$13,ZW13,ZW11,ZYW09,ZWWY08]. Our developed new model is completely different from these simulation tools. To the best of our knowledge, no existing studies have been conducted to systemically investigate the CAV traffic flow model and simulate mutually-coupled vehicle interactions. Inspired by dynamic motion patterns of real-world natural continuum systems such as the elastic solid (e.g. a piece of deformable rubber), the viscous volume (such as honey drops or oily ointments) and the fluid (like water or smoke), we formulate the mutually-coupled vehicle interactions using virtual forces. Being free of the individual randomness in agent-based methods [RNB$^+$07] and/or the limitation of one-way-coupled vehicle manipulation adopted in classic car-following models, we believe the proposed model is able to more accurately describe macroscopic traffic flow dynamics in the context of CAV-enabled traffic systems.

4.3 Methodology

In this section, we first explain how to formulate mutually-coupled vehicle interactions by the virtual forces. Based on it, the equation of motion of an elementary particle on the traffic can be formulated as the force equilibrium. This equation is aggregated to model the macroscopic traffic behavior. To do so, we have to resort to numerical computations, and we choose to use SPH in our paper. After the density and velocity at each particle is computed, the complete information of the CAV-enabled traffic is obtained.

4.3.1 Virtual Forces Driven Traffic Flow Model

We consider the CAV-enabled traffic as a 2D homogeneous dynamic continuum system, wherein all CAVs have identical operational characteristics. Sharing a similar spirit of other continuum traffic models' development [Zha02, LPMS13], our model is based on an analytical formulation of several external and internal *virtual forces* that allow mutually-coupled vehicle-vehicle and vehicle-environment interactions. In the remaining part of the paper, we use a bold lower-case letter like $\mathbf{v}$ or $\mathbf{u}$ to denote a vector-value quantity, and a regular letter like v or u to denote a scalar quantity.

Virtual external forces We explicitly define an external force that drives the traffic motion. Analogous to the gravity force, this force induces a constant acceleration field $\mathbf{g}$ over the entire traffic heading to the destination. Should the roadway be curved, the acceleration always aligns with the roadway's tangent. As to be detailed in Section 4.4.2, we use the Bézier spline (with parameter p) to describe the roadway geometry, and the tangent vector can be calculated as $[\partial x/\partial p, \partial y/\partial p]^\top \in \mathbb{R}^2$. Clearly, the traffic speed driven by $\mathbf{g}$ would approach an infinity if the roadway is sufficiently long. While such behavior maximizes the transportation capacity, it is always favored that the traffic velocity be capped by a certain limit. Therefore, we plug in another *damping force* $\mathbf{f}_d$ that penalizes an excessively high-speed traffic. We use the *proportional damping* model [Adh06] so that $\mathbf{f}_d$ is linearly proportional to the traffic speed, yet along the opposite direction: $\mathbf{f}_d = -c\mathbf{u}$, where c is the *damping factor*. For a given speed limit $\bar{\mathbf{u}}$, c can be computed as $\|\mathbf{g}\|/\|\bar{\mathbf{u}}\|$ so that $\mathbf{f}_d$ completely cancels out $\mathbf{g}$ when the speed limit is reached, and no further acceleration will be obtained. Putting together, the virtual external force at an arbitrary location on the traffic with density ρ is defined as:

$$\mathbf{f}_{ext} = \rho(\mathbf{g} - c\mathbf{u}). \tag{4.1}$$

Virtual internal forces The virtual internal forces are devised to model the between-vehicle interaction in a CAV-enabled traffic based on the assumption that a CAV is aware of the traffic condition over a wider surrounding neighborhood.

Intuitively, vehicles tend to move from a high density area to a low density area[1], and a *pressure force* $\mathbf{f}_p$ is the resultant of such non-uniform density distribution following the negative direction of the density gradient. To incorporate various fundamental diagrams, we formulate this force as:

$$\mathbf{f}_p = -\nabla(k\rho^\gamma), \tag{4.2}$$

where k and γ are two positive constants related to a desired fundamental diagram.

Similar to $\mathbf{f}_p$, the *viscosity force* $\mathbf{f}_v$ intends to make the velocity distribution uniform, i.e. a vehicle will accelerate when its neighbors are moving faster, or decelerate when its neighbors have lower velocity. Since velocity is essentially a vector field, we model $\mathbf{f}_v$ using the *Laplacian* defined as the divergence of the velocity gradient:

$$\mathbf{f}_v = \nabla^2(\mu\mathbf{u}), \tag{4.3}$$

[1]Here the density is a *local* concept. For example, a vehicle tends to move from a congested lane to the adjacent lane with less traveling vehicles. One should not confuse it with the global traffic distribution in the entire city.

where μ is a constant coefficient. We note that $\mathbf{f}_p$ and $\mathbf{f}_v$ are two-dimensional vectors. Their components perpendicular to the current traffic direction allow the lane-changing behavior. If lane A has a lower vehicle density than lane B, vehicles on lane B are likely to switch to lane A due to the existence of $\mathbf{f}_p$. Likewise, if lane A is a high-speed lane, it is also more attractive to the traffic due to $\mathbf{f}_v$.

Lastly, the equation of motion of an elementary volume in the traffic can be written as the equilibrium of all the external and internal virtual forces:

$$\rho \frac{D\mathbf{u}}{Dt} = \rho(\mathbf{g} - c\mathbf{u}) - \nabla(k\rho^{\gamma}) + \nabla^2(\mu\mathbf{u}), \tag{4.4}$$

where $\frac{D}{Dt}$ is the material derivative $\frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla$. Equation (4.4) is essentially a reiteration of the Newton's second law of motion. The left hand side of the equation is the inertial force (i.e. the mass times the acceleration) which should be balanced by all of the other forces being exerted.

Discussion It can be seen that Equation (4.4) has a similar form of the *Navier-Stokes equation* [Tem84]:

$$\frac{\partial \mathbf{u}}{\partial t} + \underbrace{(\mathbf{u} \cdot \nabla)\mathbf{u}}_{\text{Convection}} - \underbrace{\nu \nabla^2 \mathbf{u}}_{\text{Fluid viscosity}} = \underbrace{-\nabla \omega}_{\text{Thermodynamics}} + \mathbf{g}. \tag{4.5}$$

$\mathbf{f}_v$ in Equation (4.3) is our counterpart of the fluid viscosity and $\mathbf{f}_p = -\nabla(k\rho^{\gamma})$ is analogous to $-\nabla\omega$, the thermodynamic force. To satisfy the traffic volume conservation law, we employ the equivalent mass conservation constraint:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho\mathbf{u}) = 0. \tag{4.6}$$

This constraint can be automatically satisfied in a Lagrangian method such as SPH (which is discussed in the following section), because all the differential volumes in the traffic are always tracked and computed.

4.3.2 Discretization and SPH

We discretize the continuous traffic flow with a particle system similar to the fluid simulation in [MCG03]. One should note that a particle represents a small volume of the traffic flow and is not necessarily equivalent to a real vehicle. A particle carries many useful quantities such as the positional coordinates, the density, the acceleration, the velocity, and the id of the road where it is traveling. The state of the aggregated particles determines the entire traffic flow, i.e. the distribution of the density and velocity field.

SPH was invented to simulate astrophysical phenomena by Lucy [Luc77] and Gingold and Monaghan [GM77] in 1977. It is essentially an interpolation method for a particle system [Mon92, LL10]. Unlike other numerical methods, for instance the well known finite difference method (FDM), SPH does not need a grid to evaluate the spatial derivatives. Instead, SPH uses the differentiation of the interpolation formula directly. This simplifies the computation. Although SPH is less accurate than FDM, it is general enough and has been adapted into the Computer Graphics community [SF95, MCG03, BEG11] for simulating various physical phenomena, such as water, lava and deformable soft bodies. To have more details, we refer the interested readers to a wide-ranging list of the SPH papers (e.g. [Luc77, GM77, Mon92, LL10, SF95, MCG03, BEG11]).

The key idea in SPH is to use a non-negative smoothing kernel function W such that $\int W(\mathbf{r}-\mathbf{r}_0, h)\mathrm{d}\mathbf{r} = 1$ to approximate Dirac delta function δ: $\lim_{h\to 0} W(\mathbf{r}-\mathbf{r}_0, h) = \delta(\mathbf{r}-\mathbf{r}_0)$. As a result, the value of an integrable smooth function $A(\mathbf{r})$ at $\mathbf{r}_0$ can be approximated as:

$$A(\mathbf{r}_0) = \int A(\mathbf{r})\delta(\mathbf{r}-\mathbf{r}_0)\mathrm{d}\mathbf{r} \approx \int A(\mathbf{r})W(\mathbf{r}-\mathbf{r}_0, h)\mathrm{d}\mathbf{r}, \tag{4.7}$$

where $h > 0$ is the support radius. When the continuous function domain is discretized using a particle system, the integral in Equation (4.7) is also discretized as

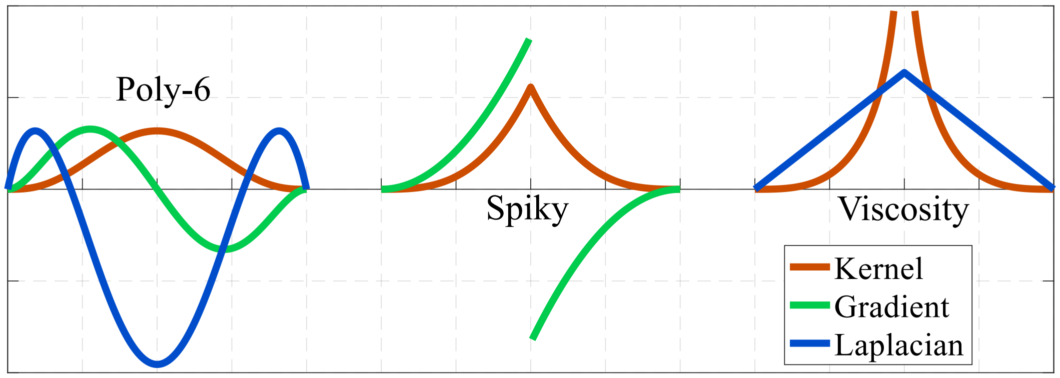


Figure 4.1: We use the 2D poly-6, spiky and viscosity kernel for the SPH simulation. The spiky kernel is for the gradient evaluation needed for $\mathbf{f}_p$. The viscosity kernel is for the Laplacian evaluation of $\mathbf{f}_v$.

a summation:

$$A(\mathbf{r}) \approx \sum_{j \in \mathcal{N}} A_j \frac{m_j}{\rho_j} W(\mathbf{r}_j - \mathbf{r}, h), \tag{4.8}$$

where $\mathcal{N}$ is the set of indices of the neighboring particles. Derivatives of $A(\mathbf{r})$ can be approximated in a similar way:

$$\nabla A(\mathbf{r}) \approx \sum_{j \in \mathcal{N}} A_j \frac{m_j}{\rho_j} \nabla W(\mathbf{r}_j - \mathbf{r}, h). \tag{4.9}$$

4.3.3 Numerical Simulation and Smoothing Kernels

The Gaussian kernel is a widely-used choice for smoothing kernel function W. However, the Gaussian kernel does not have a compact support (i.e. the support radius of the Gaussian is infinite), and it is expensive to evaluate. We follow the suggestions in [MCG03], and employ three 2D polynomial smoothing kernels namely the *poly-6 kernel*, the *spiky kernel*, and the *viscosity kernel* to numerically compute necessary physical and kinematic quantities as shown in Fig. 4.1:

$$\begin{cases} W_{poly6}(\mathbf{r},h) &= \dfrac{4}{\pi h^8}(h^2-\|\mathbf{r}\|^2)^3 \\ W_{spiky}(\mathbf{r},h) &= \dfrac{10}{\pi h^5}(h-\|\mathbf{r}\|)^3 \\ W_{viscosity}(\mathbf{r},h) &= \dfrac{40}{\pi h^2}\Big[-\dfrac{\|\mathbf{r}\|^3}{9h^3}+\dfrac{\|\mathbf{r}\|^2}{4h^2}-\dfrac{1}{6}\ln\Big(\dfrac{\|\mathbf{r}\|}{h}\Big)-\dfrac{5}{36}\Big] \end{cases}$$

The poly-6 kernel possesses a Gaussian-like bell curve with a compact support, and it is easy to evaluate. It serves as the default kernel in our model. To evaluate $\mathbf{f}_p$ however, the poly-6 kernel is not suitable because its gradient gets to zero as $\|\mathbf{r}\| \to 0$ (i.e. see Fig. 4.1). This property implies that when particles approach each other, $\mathbf{f}_p$ becomes smaller and the particles tend to get clustered. This is not desired in a traffic flow. As a result, the spiky kernel with a high gradient value as $\|\mathbf{r}\| \to 0$ is used for computing $\mathbf{f}_p$. For a similar reason, we use the viscosity kernel for $\mathbf{f}_v$. The Laplacian of the viscosity kernel has a cone shape with its maximum at $\|\mathbf{r}\| = 0$, and it is positive within the support area which is required for the computation. All of these three kernels have vanished values when $\|\mathbf{r}\| > h$. Our experiments show that the three kernels work well in our traffic simulation.

The gradient and Laplacian operators as appeared in Equations (4.2) and (4.3) for the spiky and viscosity kernels can be evaluated as:

$$\begin{cases} \nabla W_{spiky}(\mathbf{r},h) &= -\dfrac{30}{\pi h^5}(h-\|\mathbf{r}\|)^2\dfrac{\mathbf{r}}{\|\mathbf{r}\|} \\ \nabla^2 W_{viscosity}(\mathbf{r},h) &= \dfrac{40}{\pi h^5}(h-\|\mathbf{r}\|). \end{cases} \tag{4.10}$$

Putting all together, ρ, $\mathbf{f}_p$, and $\mathbf{f}_v$ at the i^{th} particle can be computed as:

$$\begin{aligned} \rho_i &= \sum_{j\in\mathcal{N}} m_j W_{poly6}(\mathbf{r}_j-\mathbf{r}_i,h) \\ \mathbf{f}_{p_i} &= -k\sum_{j\in\mathcal{N}} \frac{\rho_i^\gamma+\rho_j^\gamma}{2}\frac{m_j}{\rho_j}\nabla W_{spiky}(\mathbf{r}_j-\mathbf{r}_i,h) \\ \mathbf{f}_{v_i} &= \mu\sum_{j\in\mathcal{N}}(\mathbf{u}_j-\mathbf{u}_i)\frac{m_j}{\rho_j}\nabla^2 W_{viscosity}(\mathbf{r}_j-\mathbf{r}_i,h). \end{aligned} \tag{4.11}$$

Noting that SPH is a Lagrangian-style method, and the convection term in Equation (4.5) is implicitly incorporated in our model. The unknown density ρ, as well as the associated virtual forces $\mathbf{f}_p$ and $\mathbf{f}_v$ (assuming $\mathbf{g}$ is known) are calculated using Equation (4.11). Finally, the velocities and the new positions for the particles are computed using numerical time integration, and we are ready to simulate the traffic condition of the next time instance.

4.3.4 Incorporating Existing Traffic Flow Models

So far, we have discussed a continuum model for CAV-enabled traffic, which uses two-dimensional internal and external forces for modeling the mutually-coupled CAV interactions. Most existing macroscopic traffic flow models based on conventional traffic are one-dimensional and incorporate various fundamental diagrams. A fundamental diagram is determined by the relationship between the density and velocity. For instance, the classic Greenshields model [RC02] presents a linear relationship between the density and velocity. In this subsection, we show that our model is versatile, and it can naturally be specialized to the conventional traffic. Therefore, the proposed traffic model could be considered a *generalization* of the existing traffic flow theory.

We downgrade our model to one-dimensional, and Equation (4.4) becomes:

$$\rho\frac{\partial u}{\partial t} = -k\frac{\partial(\rho^\gamma)}{\partial x} + \mu\frac{\partial^2 u}{\partial x^2} + \rho(g - cu). \tag{4.12}$$

If we consider a traffic equilibrium under a uniform velocity u_0 (f_v becomes zero in this case), without the external force, Equation (4.12) can be simplified as:

$$\rho\frac{\partial u}{\partial t} = -k\gamma\rho^{\gamma-1}\frac{\partial \rho}{\partial x}. \tag{4.13}$$

By the traffic volume conservation law (i.e. $\frac{\partial \rho}{\partial t} + u_0\frac{\partial \rho}{\partial x} = 0$), we have:

$$\frac{\partial \rho}{\partial x} = -\frac{1}{u_0}\frac{\partial \rho}{\partial t}. \tag{4.14}$$

Combining Equations (4.13) and (4.14) yields:

$$\frac{\partial u}{\partial t} = k\gamma\rho^{\gamma-2}\frac{1}{u_0}\frac{\partial \rho}{\partial t}.$$

Integrating both sides of the above equation leads to:

$$u = \begin{cases} \dfrac{k}{u_0}\ln\rho + C & \gamma = 1 \\ \dfrac{k\gamma}{u_0(\gamma-1)}\rho^{\gamma-1} + C & \text{otherwise,} \end{cases}$$

where C is a constant. As mentioned previously, k and γ are parameters related to the desired traffic fundamental diagram. When we take $\gamma = 2$, we obtain the linear regression relation $u = \frac{2k}{u_0}\rho + C$, which is the classic Greenshields model [RC02]. Other fundamental diagrams, especially in polynomial relationships between the density and velocity, could be derived similarly by adjusting parameters k and γ.

The SPH-based simulation can also be easily degenerated to 1D, and the 1D poly-6 kernel is:

$$W_{poly6}(x,h) = \frac{35}{32h^7}(h^2 - x^2)^3.$$

Note that this formulation is slightly different from its 2D counterpart. This is because the partition of unity condition must be satisfied (i.e. $\int W(\mathbf{r}-\mathbf{r}_0,h)d\mathbf{r} = 1$).

In the conventional traffic, a vehicle only interacts with its leading vehicle. As a result, *we only use the positive half of the kernel function* when computing f_p and f_v. That is the summation is only over the particles in front of the current particle. The gradient and Laplacian of the 1D spiky and viscosity kernel are:

$$\begin{aligned} \frac{\partial W_{spiky}}{\partial x} &= -\frac{6}{h^4}(h-x)^2 \\ \frac{\partial^2 W_{viscosity}}{\partial x^2} &= \frac{12}{h^4}(h-x). \end{aligned}$$

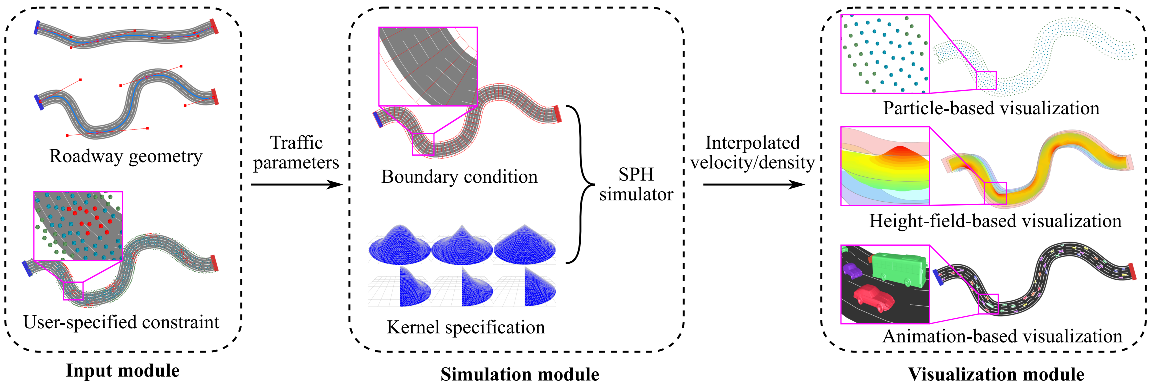

Roadway geometry
User-specified constraint
Input module
Traffic parameters
Boundary condition
Kernel specification
SPH simulator
Simulation module
Interpolated velocity/density
Particle-based visualization
Height-field-based visualization
Animation-based visualization
Visualization module

Figure 4.2: A high-level flow chart of the proposed CAV-enabled traffic simulation framework. Our system has three major modules namely the input module, the simulation module, and the visualization module. The input module provides an interface for the user to fully specify the configurations of the traffic to be simulated. This information is passed to the simulation module, which will choose necessary boundary conditions and kernel functions (according to user's specification) and run a SPH-based simulation. The output will be leveraged in the visualization module to deliver the final visual representations of the simulated traffic flow.

4.4 System Implementation

Based on the derived traffic model and the SPH simulation, we develop an interactive visual traffic simulation system. Our system is implemented using `Microsoft Visual C++` on a desktop `Windows 10` PC with `Intel Xeon E5-2670 CPU` and 32G on-board memory. The traffic simulator was developed under `Qt` software development environment with the visualization based on `OpenGL`. `OpenGL` is essentially an application programming interface (API) for computer graphics. It is independent of the operating system and hardware, providing an easy way to manipulate the graphics processing unit (GPU) for rendering arbitrary 2D or 3D graphics.

4.4.1 System Overview

Our CAV-enabled traffic simulation platform consists of three modules as outlined in Fig. 4.2. In the *input module*, the user specifies necessary traffic parameters (such as the expected traffic volume, traffic flow rates, and speed limits), roadway geometries and external constraints. External constraints are prescribed configurations of the traffic. For instance, users can force the velocity of the traffic at a given region to be zero to study the shockwave propagation, or they can block two out of four lanes of the roadway to test the resulting traffic congestion. The *simulation module* sets up the corresponding boundary conditions and kernel functions. We employ a full-kernel specification for CAV-enabled traffic simulation and a half-kernel specification for conventional traffic simulation. The core component of the simulation module is the SPH simulator. At each time step, the densities and velocities are calculated at all the particles. The continuous density and velocity field can then be interpolated. The *visualization module* uses the interpolated data obtained from the simulation module to animate and visualize the traffic flow.

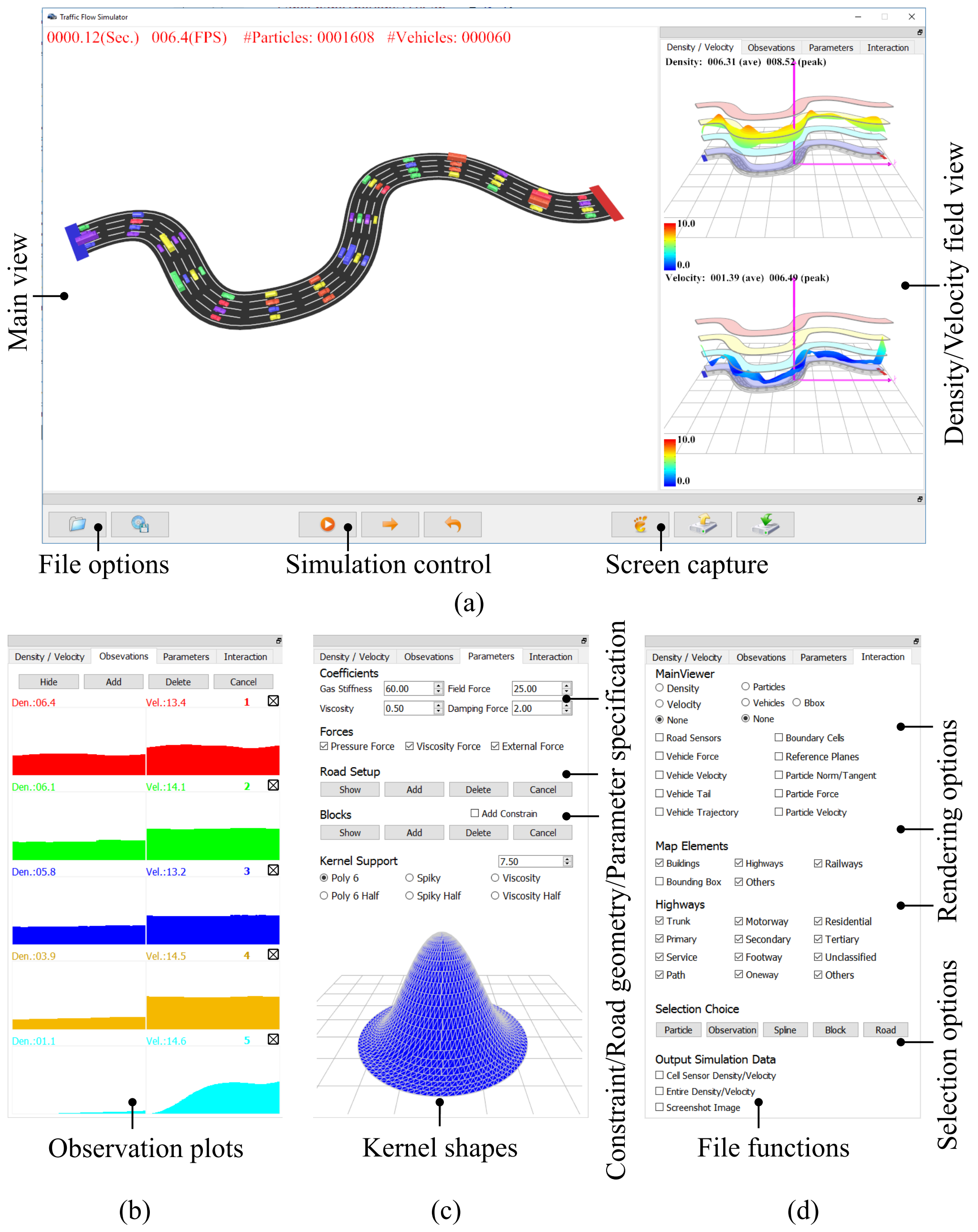
Our system provides three levels of detail for the visualization. The user is able to view the simulated traffic as a discretized particle system. The density/velocity distribution can also be rendered as a height field in 3D with a customizable color map. The user can further zoom in the traffic to see animated vehicles. This detailed microscopic traffic visualization could be important and useful in applications like virtual reality or gaming. We create a database consisting of various 3D vehicle models (vans, sedans, trucks, etc.), then randomly pick the models and scatter them over the simulated traffic flow following the density distribution, so that more vehicles are rendered and animated at high-density regions. At the starting end of the road, the vehicles are generated according to the user-specified traffic flow rate (with the unit of vehicle per hour). The traffic flow rate is also converted to the boundary condition for generating particles for the SPH simulation (e.g. a vehicle is equivalent to ten particles). In this way, the traffic volume represented by the particles is consistent with the 3D vehicle models. Based on the position within the traffic at each animation frame, the i^{th} vehicle will have a velocity $\mathbf{u}_i$ according to the SPH simulation, and its position $\mathbf{x}_i$ for the next frame will be updated as $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{u}_i$. Here Δt denotes the time interval between two consecutive animation frames.

The graphical user interface (GUI) of our implemented system is shown in Fig. 4.3. It contains a main interface, four tabs, and other simulation controls. Next to the main visualization panel, there are two sub-windows for the density/velocity field visualization on the right. The second tab (Fig. 4.3 (b)) reports the density/velocity plots at positions of interest in the traffic, which can be interactively specified by the user. The third and fourth tabs provide miscellaneous functions/options for interactive settings, such as roadway geometries, constraints, kernels, forces, rendering options, and file I/O, as shown in Fig. 4.3 (c) and (d).

4.4.2 Roadway Geometries

To build up a realistic roadway segment, we use a cubic Bézier spline to model the roadway geometry variation at its center line (known as the neutral axis too). The Bézier spline is a sequence of piece-wise cubic polynomial curves that are connected smoothly. We further extend the roadway segment to 2D by specifying the vertical span of the splines along the normal direction (i.e. the direction perpendicular to the tangent of the splines or the traffic direction). With this method, we can model various roadways with arbitrary shapes and widths (see the input module in Fig. 4.2).

A roadway segment is further divided into a few polygons or *cells*. All cells together form the simulation domain of the roadway segment (see the simulation module in Fig. 4.2). We set the cell to be wider than the roadway segment by two support



0000.12(Sec.)   006.4(FPS)   #Particles: 0001608   #Vehicles: 000060

Density: 006.31 (ave) 008.52 (peak)

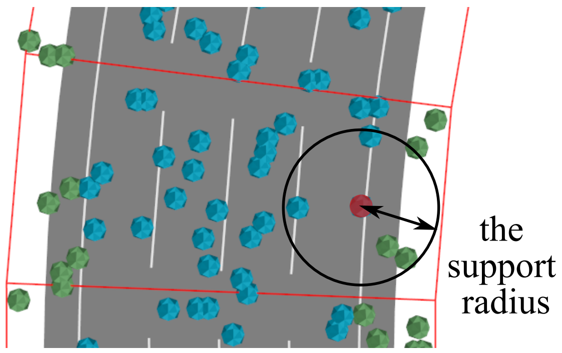
Velocity: 001.39 (ave) 006.49 (peak)

Den.:06.4 Vel.:13.4 1
Den.:06.1 Vel.:14.1 2
Den.:05.8 Vel.:13.2 3
Den.:03.9 Vel.:14.5 4
Den.:01.1 Vel.:14.6 5

Coefficients
Gas Stiffness 60.00   Field Force 25.00
Viscosity 0.50   Damping Force 2.00
Forces
☑ Pressure Force ☑ Viscosity Force ☑ External Force
Road Setup
Blocks ☐ Add Constrain
Kernel Support 7.50
◉ Poly 6 ○ Spiky ○ Viscosity
○ Poly 6 Half ○ Spiky Half ○ Viscosity Half

MainViewer
○ Density ○ Particles
○ Velocity ○ Vehicles ○ Bbox
◉ None ◉ None
☐ Road Sensors ☐ Boundary Cells
☐ Vehicle Force ☐ Reference Planes
☐ Vehicle Velocity ☐ Particle Norm/Tangent
☐ Vehicle Tail ☐ Particle Force
☐ Vehicle Trajectory ☐ Particle Velocity
Map Elements
☑ Buildings ☑ Highways ☑ Railways
☐ Bounding Box ☑ Others
Highways
☑ Trunk ☑ Motorway ☑ Residential
☑ Primary ☑ Secondary ☑ Tertiary
☑ Service ☑ Footway ☑ Unclassified
☑ Path ☑ Oneway ☑ Others
Selection Choice
Particle Observation Spline Block Road
Output Simulation Data
☐ Cell Sensor Density/Velocity
☐ Entire Density/Velocity
☐ Screenshot Image

Figure 4.3: The snapshot of the graphical user interface of the implemented system. Besides the major 3D visualization window, our system provides multiple mutually-linked plots, statistics and visualization sub-views. Most of them can be interactively controlled by the user to provide the descried visual perception of the traffic.



radii, so that the particles along the roadway boundary can have a full support. This treatment smooths the boundary of the resulting density/velocity field. The particles outside of the roadway boundary serve similarly as the so-called ghost particles, which is a popular method in SPH simulation for boundary treatment [FAMO99, Fed02, SB12]. We pre-compute and build a query table, whose entries are indices of the cells. The corresponding context includes many useful attributes associated with the cell, such as the road id, positional coordinates, tangent and normal vectors. The cells can also be used for fast boundary treatment. We employ the explicit project-and-reflect method: the particle is simply projected to the boundary where it is closest to once it moves out of the boundary [CR99]. With the cell-based subdivision, the roadway boundary becomes a piece-wise line segment, and it is trivial to project out-of-road particles back onto the boundary.

4.4.3 Simulation in Roadway Networks

A roadway network can be described as a collection of roadway segments connected by intersections. We divide the entire traffic flow in the roadway network into many small pieces. Each piece is in fact the aggregation of all vehicles with the similar trajectories. The user-specified trajectory data may come from real-world traffic or predicted traffic scenarios. The trajectories then determine the simulation domain associated with each piece which usually consists of a set of connected roadway segments. The simulation result is the linear combination of the simulation of all the pieces. Fig. 4.4 illustrates two traffic flow pieces and their associated simulation domains. In the overlapping domain, the simulation result is the linear combination

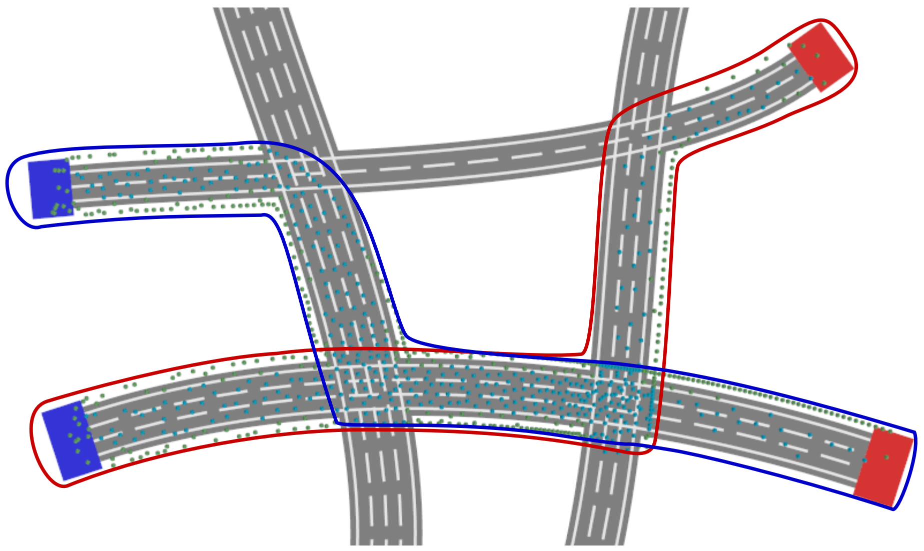


Figure 4.4: Two traffic flow pieces are simulated in their associated simulation domains (highlighted in the blue and red frame). In the overlapping domain, the simulation result is the linear combination of these two pieces.

of these two pieces.

In the conventional traffic, the intersection configurations are converted to the boundary conditions, for example, a red traffic signal phase imposes the zero-velocity constraint in the area. In the future CAV-enabled traffic, the traffic signals could be potentially removed. No additional constraints are applied to the intersection, and the traffic flow becomes uninterrupted in the entire roadway network. Our model works well in both of these two scenarios.

4.4.4 Fast Neighbor Search

The most expensive computation along the simulation is the summation over the neighborhood for all the particles (i.e. Equations (4.8) and (4.9)). Because particles are moving along the simulation, each particle's neighbors within the support radius also vary at each time step. A brutal force iteration overall the particles

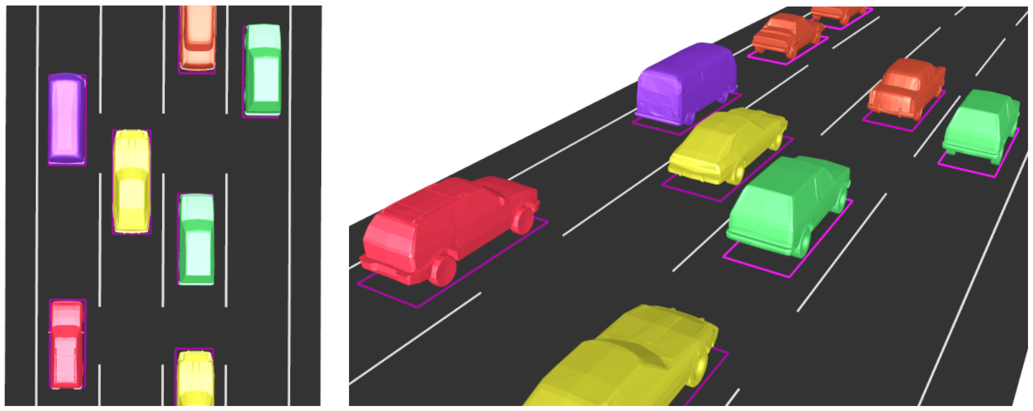


Figure 4.5: The detailed traffic visualization uses animated 3D vehicle models to provide users an intuitive impression of the traffic. Each vehicle possesses a bounding box (the purple rectangle), which will be used for collision detection and handling.

leads to a $\mathbf{O}(N^2)$ calculation which is expensive noting that it is only for a single animation frame. To accelerate this procedure, we employ the spatial hashing method [THM+03] to fast retrieve the information of neighboring particles. Therefore, the time complexity of the simulation becomes linear with respect to the total number of particles.

4.4.5 Collision Detection and Handling

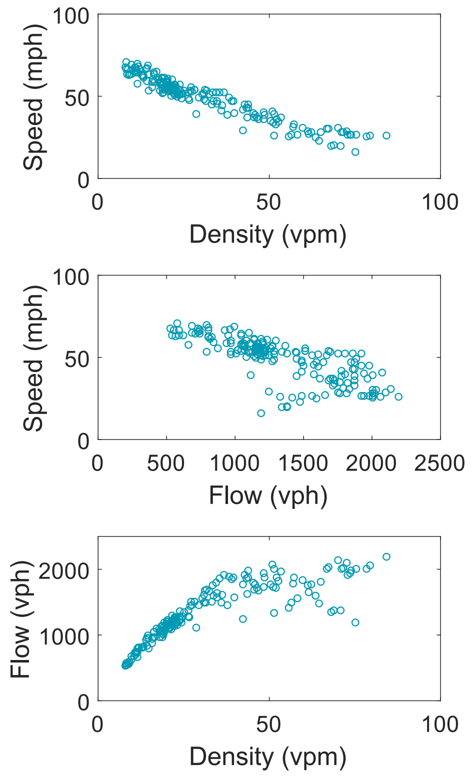
In the microscopic zoomed-in traffic visualization, we use animated 3D vehicle models to represent the traffic detail as shown in Fig. 4.5. Each rendered vehicle follows the position-dependant velocity obtained from the simulation. Because our force-based traffic model is an aggregated one, and it does not account for the microscopic vehicle behavior, it is possible that the animated vehicles collide each other. The collision detection is achieved by checking the overlapping region between 2D boundary boxes of the 3D vehicle models. Once a collision is detected, we roll back the simulation and apply a virtual *penalty force* to push the colliding vehicles away. It can be understood as connecting a vehicle with its neighbors by invisible springs, and the spring force will be triggered if a collision is detected and pulling the involved vehicles back.

4.5 Experimental Results

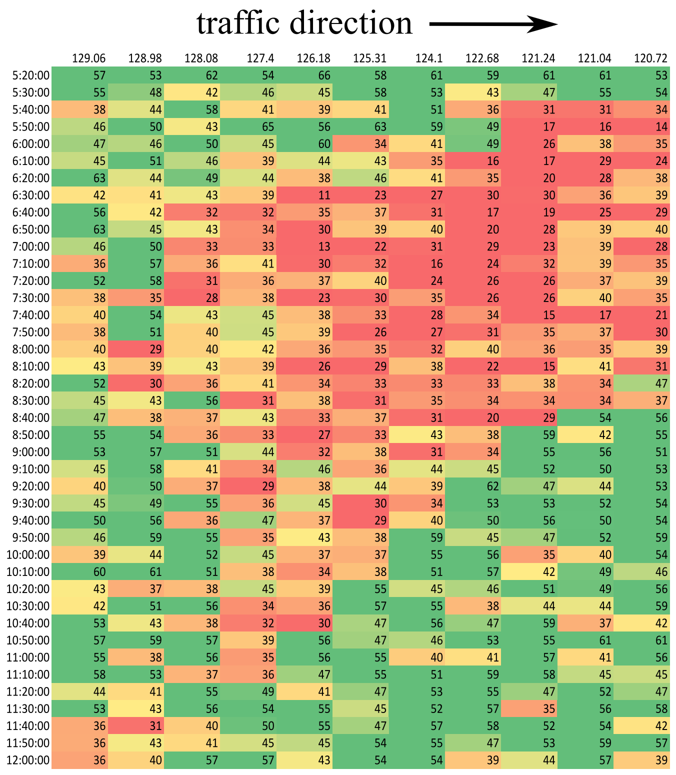
The proposed virtual-force-driven traffic model and the implemented visual simulation system have been extensively tested. Since CAV technique has not yet been widely deployed, we downgrade our model to the classic Greenshields model (i.e. as discussed in Section 4.3.4), and then validate the model by calibrating the simulation result with the real-world traffic data. Afterwards, we simulate the CAV-enable traffic under various scenarios by imposing user-specified external constraints. *Please also refer to the accompanying video for more details.*

4.5.1 Model Validation

First, we show that our simulation based on the downgraded model well matches the real-world traffic observation. The ground-truth data are collected from the 11 inductance loop detector stations along `State Route 520` in the `greater Seattle areas` in the `State of Washington`. These loop detector stations are selected because this roadway segment is one major corridor crossing the `Lake of Washington`, and the uninterrupted traffic flow is well characterized to minimize the on-ramp and off-ramp traffic disruption. Both stationary and dynamic characteristics of the traffic flow are considered. Fig. 4.6 (a) shows the general correlation among the traffic flow, speed, and density when the traffic becomes stationary (i.e. the fundamental diagram). Fig. 4.6 (b) plots the dynamic shockwave transmission and dissipation of the congestion during the morning peak-hour time period form 5:40 to 8:30 AM. The horizontal axis is the loop station milestone and the vertical axis gives the time instant stamp.



(a)



traffic direction

	129.06	128.98	128.08	127.4	126.18	125.31	124.1	122.68	121.24	121.04	120.72
5:20:00	57	53	62	54	66	58	61	59	61	61	53
5:30:00	55	48	42	46	45	58	53	43	47	55	54
5:40:00	38	44	58	41	39	41	51	36	31	31	34
5:50:00	46	50	43	65	56	63	59	49	17	16	14
6:00:00	47	46	50	45	60	34	41	49	26	38	35
6:10:00	45	51	46	39	44	43	35	16	17	29	24
6:20:00	63	44	49	44	38	46	41	35	20	28	38
6:30:00	42	41	43	39	11	23	27	30	30	36	39
6:40:00	56	42	32	32	35	37	31	17	19	25	29
6:50:00	63	45	43	34	30	39	40	20	28	39	40
7:00:00	46	50	33	33	13	22	31	29	23	39	28
7:10:00	36	57	36	41	30	32	16	24	32	39	35
7:20:00	52	58	31	36	37	40	24	26	26	37	39
7:30:00	38	35	28	38	23	30	35	26	26	40	35
7:40:00	40	54	43	45	38	33	28	34	15	17	21
7:50:00	38	51	40	45	39	26	27	31	35	37	30
8:00:00	40	29	40	42	36	35	32	40	36	35	39
8:10:00	43	39	43	39	26	29	38	22	15	41	31
8:20:00	52	30	36	41	34	33	33	33	38	34	47
8:30:00	45	43	56	31	38	31	35	34	34	34	37
8:40:00	47	38	37	43	33	37	31	20	29	54	56
8:50:00	55	54	36	33	27	33	43	38	59	42	55
9:00:00	53	57	51	44	32	38	31	34	55	56	51
9:10:00	45	58	41	34	46	36	44	45	52	50	53
9:20:00	40	50	37	29	38	44	39	62	47	44	53
9:30:00	45	49	55	36	45	30	34	53	53	52	54
9:40:00	50	56	36	47	37	29	40	50	56	50	54
9:50:00	46	59	55	35	43	38	59	45	47	52	59
10:00:00	39	44	52	45	37	37	55	56	35	40	54
10:10:00	60	61	51	38	34	38	51	57	42	49	46
10:20:00	43	37	38	45	39	55	45	46	51	49	56
10:30:00	42	51	56	34	36	57	55	38	44	44	59
10:40:00	53	43	38	32	30	47	56	47	59	37	42
10:50:00	57	59	57	39	56	47	46	53	55	61	61
11:00:00	55	38	56	35	56	55	40	41	57	41	56
11:10:00	58	53	37	36	47	55	51	59	58	45	45
11:20:00	44	41	55	49	41	47	53	55	47	52	47
11:30:00	53	43	56	54	55	45	52	57	35	56	58
11:40:00	36	31	40	50	55	47	57	58	52	54	42
11:50:00	36	43	41	45	45	54	55	47	53	59	57
12:00:00	36	40	57	57	43	54	54	39	44	57	39

(b)

Figure 4.6: The real-world traffic data are collected on `State Route 520` in `Seattle`, `Washington`. The correlation between the flow, speed and density is given in (a), and the speed distribution is shown in (b) wherein the congestion is occurred and dissipated. The time interval between two rows is 10 minutes. Each column corresponds to a speed sensor. The red color indicates a low speed, while the green color indicates a relative high speed.

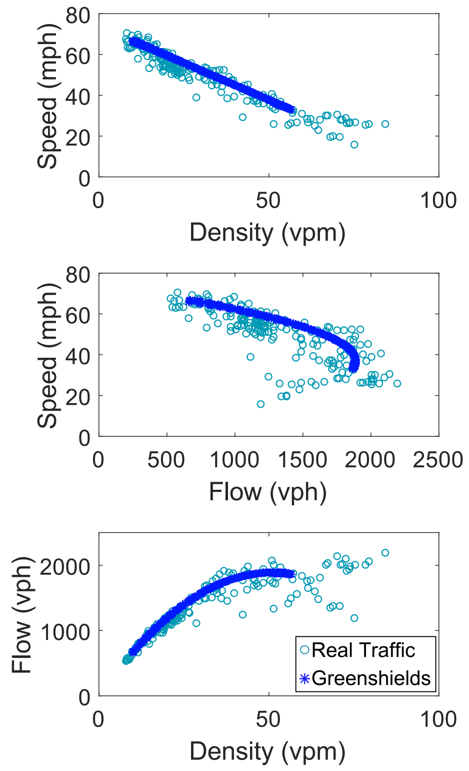
We build a one-mile roadway with 20 data observation locations as s_0 through s_{19} uniformly distributed. Assuming that an average vehicle length is 20 feet and the minimum gap between vehicles is 6 feet, the maximum vehicle density is thus around 203 vpm (vehicles per mile). We employ 2,000 particles (about 10 times of the vehicles) for simulating the traffic. The traffic flow rate gives the boundary condition at the starting end of the roadway. From the ground-truth data, we found that the traffic flow rate ranges from 500 vph (vehicles per hour) to around 2000 vph. Therefore, the corresponding rate of generating particles is around 1.4 to 5.6 particles per second in our simulation. We increase the traffic flow rate, and measure the density and the velocity after the traffic reaches a equilibrium state. The traffic and simulation parameters are reported in Tab. 4.1. Here, the Greenfield model is used by setting $\gamma = 2.0$ (i.e. see Section 4.3.4). The particle mass is a virtual quantity related to the traffic volume. We ignore the units of this quantity and other virtual-force related coefficients as well. We found the time step 0.025s is suitable for our simulation, considering the stability, efficiency and accuracy of the simulation. The performance depends on the simulation scale, e.g. it takes about 15 ms for advancing a step for the SPH simulation with 2,000 particles (with the single core implementation).

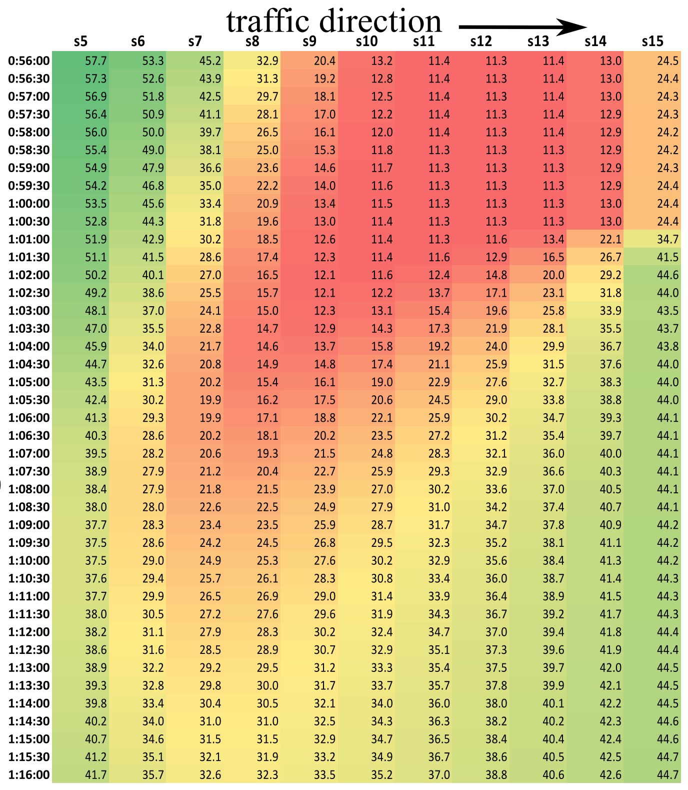
Parameters	**Values**
γ	2.0
time step Δt	0.025s
support radius h	52ft
particle mass m	40.0
pressure coefficient k	6.0
viscosity coefficient μ	1.0×10^4
external acceleration g	60.0
damping factor c	2.0

Table 4.1: Simulation parameters.

The simulation result of the fundamental diagram is reported in Fig. 4.7 (a). We also compare the congestion formation, transmission, and dissipation processes in Fig. 4.7 (b). The data at observation locations s_5 to s_{15} are presented. It can be seen that the simulated traffic appears less noisy and smoother than the real-world data, nevertheless they match each other well.

We further create an “artificial” crash event after the traffic reaches its equilibrium state and simulate how is the traffic congestion formed up and how does shockwave transmit. The traffic velocity at the accident area is constrained to be zero to mimic the real traffic operations and to form up the bottleneck. The conges-




traffic direction

	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
0:56:00	57.7	53.3	45.2	32.9	20.4	13.2	11.4	11.3	11.4	13.0	24.5
0:56:30	57.3	52.6	43.9	31.3	19.2	12.8	11.4	11.3	11.4	13.0	24.4
0:57:00	56.9	51.8	42.5	29.7	18.1	12.5	11.4	11.3	11.4	13.0	24.3
0:57:30	56.4	50.9	41.1	28.1	17.0	12.2	11.4	11.3	11.4	12.9	24.3
0:58:00	56.0	50.0	39.7	26.5	16.1	12.0	11.4	11.3	11.4	12.9	24.2
0:58:30	55.4	49.0	38.1	25.0	15.3	11.8	11.3	11.3	11.3	12.9	24.2
0:59:00	54.9	47.9	36.6	23.6	14.6	11.7	11.3	11.3	11.3	12.9	24.3
0:59:30	54.2	46.8	35.0	22.2	14.0	11.6	11.3	11.3	11.3	12.9	24.4
1:00:00	53.5	45.6	33.4	20.9	13.4	11.5	11.3	11.3	11.3	13.0	24.4
1:00:30	52.8	44.3	31.8	19.6	13.0	11.4	11.3	11.3	11.3	13.0	24.4
1:01:00	51.9	42.9	30.2	18.5	12.6	11.4	11.3	11.6	13.4	22.1	34.7
1:01:30	51.1	41.5	28.6	17.4	12.3	11.4	11.6	12.9	16.5	26.7	41.5
1:02:00	50.2	40.1	27.0	16.5	12.1	11.6	12.4	14.8	20.0	29.2	44.6
1:02:30	49.2	38.6	25.5	15.7	12.1	12.2	13.7	17.1	23.1	31.8	44.0
1:03:00	48.1	37.0	24.1	15.0	12.3	13.1	15.4	19.6	25.8	33.9	43.5
1:03:30	47.0	35.5	22.8	14.7	12.9	14.3	17.3	21.9	28.1	35.5	43.7
1:04:00	45.9	34.0	21.7	14.6	13.7	15.8	19.2	24.0	29.9	36.7	43.8
1:04:30	44.7	32.6	20.8	14.9	14.8	17.4	21.1	25.9	31.5	37.6	44.0
1:05:00	43.5	31.3	20.2	15.4	16.1	19.0	22.9	27.6	32.7	38.3	44.0
1:05:30	42.4	30.2	19.9	16.2	17.5	20.6	24.5	29.0	33.8	38.8	44.0
1:06:00	41.3	29.3	19.9	17.1	18.8	22.1	25.9	30.2	34.7	39.3	44.1
1:06:30	40.3	28.6	20.2	18.1	20.2	23.5	27.2	31.2	35.4	39.7	44.1
1:07:00	39.5	28.2	20.6	19.3	21.5	24.8	28.3	32.1	36.0	40.0	44.1
1:07:30	38.9	27.9	21.2	20.4	22.7	25.9	29.3	32.9	36.6	40.3	44.1
1:08:00	38.4	27.9	21.8	21.5	23.9	27.0	30.2	33.6	37.0	40.5	44.1
1:08:30	38.0	28.0	22.6	22.5	24.9	27.9	31.0	34.2	37.4	40.7	44.1
1:09:00	37.7	28.3	23.4	23.5	25.9	28.7	31.7	34.7	37.8	40.9	44.2
1:09:30	37.5	28.6	24.2	24.5	26.8	29.5	32.3	35.2	38.1	41.1	44.2
1:10:00	37.5	29.0	24.9	25.3	27.6	30.2	32.9	35.6	38.4	41.3	44.2
1:10:30	37.6	29.4	25.7	26.1	28.3	30.8	33.4	36.0	38.7	41.4	44.3
1:11:00	37.7	29.9	26.5	26.9	29.0	31.4	33.9	36.4	38.9	41.5	44.3
1:11:30	38.0	30.5	27.2	27.6	29.6	31.9	34.3	36.7	39.2	41.7	44.3
1:12:00	38.2	31.1	27.9	28.3	30.2	32.4	34.7	37.0	39.4	41.8	44.4
1:12:30	38.6	31.6	28.5	28.9	30.7	32.9	35.1	37.3	39.6	41.9	44.4
1:13:00	38.9	32.2	29.2	29.5	31.2	33.3	35.4	37.5	39.7	42.0	44.5
1:13:30	39.3	32.8	29.8	30.0	31.7	33.7	35.7	37.8	39.9	42.1	44.5
1:14:00	39.8	33.4	30.4	30.5	32.1	34.0	36.0	38.0	40.1	42.2	44.5
1:14:30	40.2	34.0	31.0	31.0	32.5	34.3	36.3	38.2	40.2	42.3	44.6
1:15:00	40.7	34.6	31.5	31.5	32.9	34.7	36.5	38.4	40.4	42.4	44.6
1:15:30	41.2	35.1	32.1	31.9	33.2	34.9	36.7	38.6	40.5	42.5	44.7
1:16:00	41.7	35.7	32.6	32.3	33.5	35.2	37.0	38.8	40.6	42.6	44.7

(a) (b)

Figure 4.7: The simulation results obtained from our downgraded model (i.e. the classic Greenshields model) well match the real-world observation.

tion spreads out and the shockwave is transmitted backward from the high-density area to the low one. Eventually, after the crash is fully cleared up and the capacity is restored, the density distribution becomes uniform as the same as the initial state. The results are shown in Fig. 4.8.

4.5.2 CAV-enabled Traffic Simulation

CAV-enabled traffic simulation is conducted using the original two-dimensional model with the standard full-kernel setting for the SPH simulator. The new fundamental diagram is shown in Fig. 4.9 (a), where we note that the speed is in fact the magni-

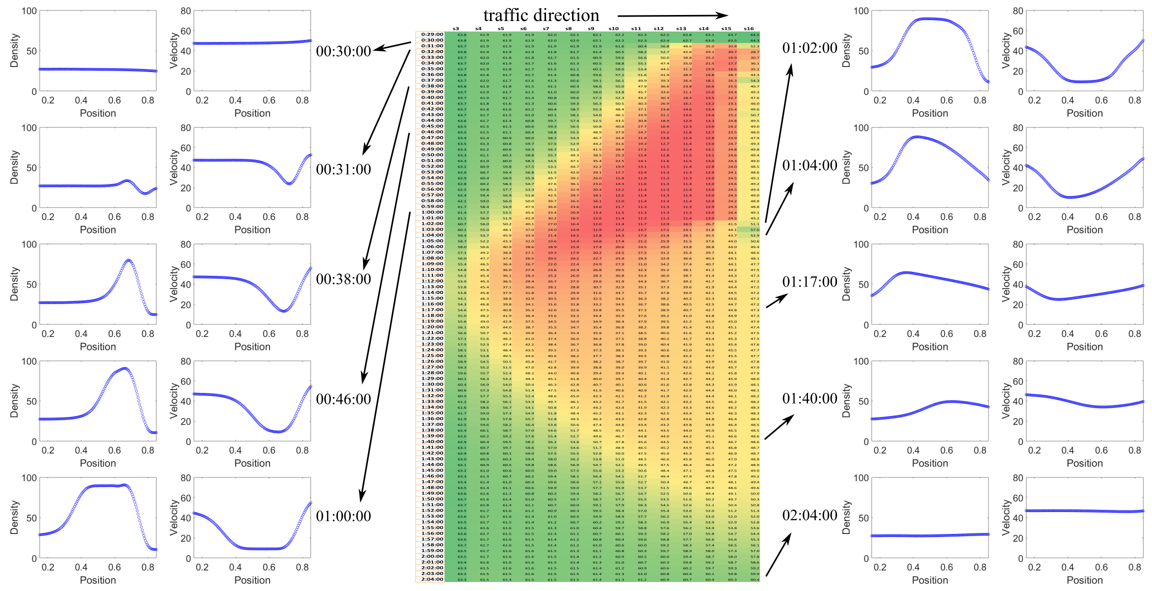
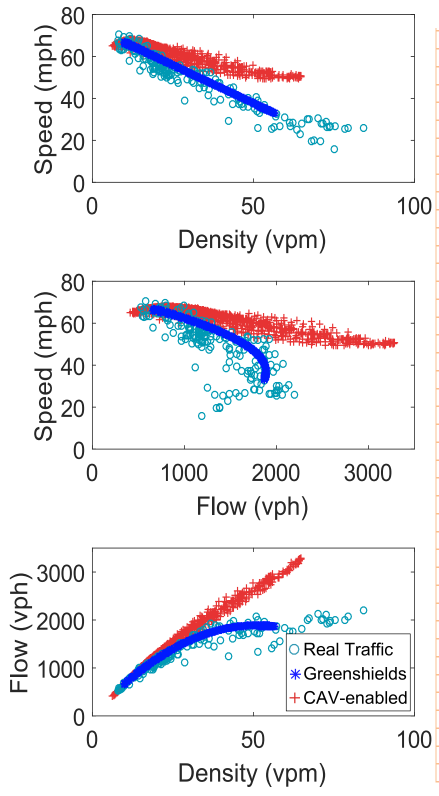


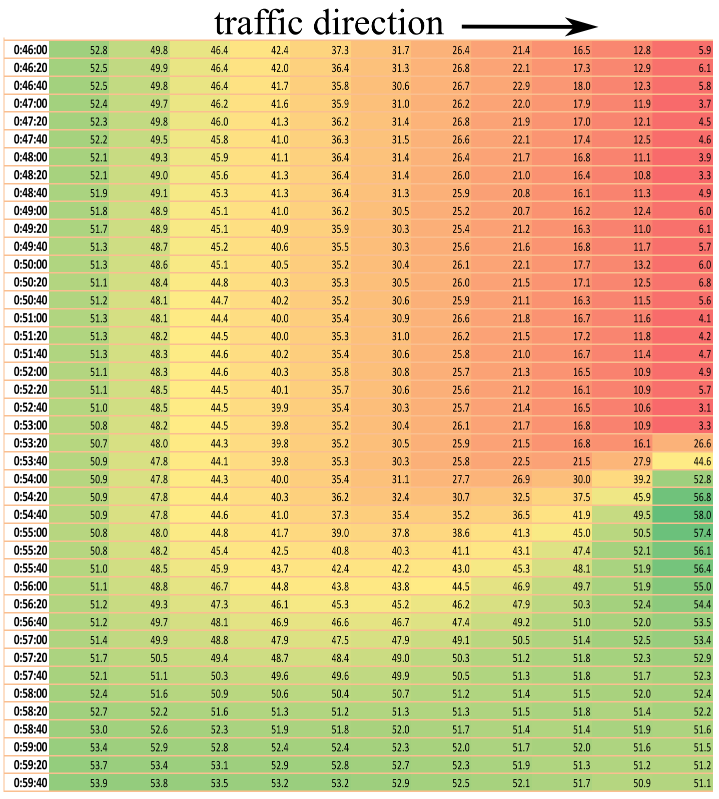
Figure 4.8: The simulation results using our downgraded 1D model under an artificial congestion scenario are presented. We constrain the traffic velocity at the accident area to be zero. The resulting traffic congestion is visualized. The accident is cleared after 30 minutes and the traffic capacity is restored eventually.

tude of the velocity vector obtained from the simulation. We can see that the CAV traffic flow is characterized by a quite different fundamental diagram compared to the traditional traffic flow model (Fig. 4.7 (a)). Rather than a straight line (one-to-one matching) characterized by the Greenshields model, the CAV speed-density relationship shows its over-dispersed distribution with all many-to-many matching patterns. One single speed may correspond to multiple possible density states due to the two-way communication and collaborative linkages among CAVs. Similar results are observed in the flow-speed and flow-density relationships. For example, rather than a parabolic curve, the relationship between the flow and density tends to spread out as an area-wide distribution. Additionally, we can see that the traffic capacity is significantly improved in the CAV-enabled traffic as expected.

Fig. 4.9 (b) reports a distinctive congestion formation and dissipation pattern for the CAV-enabled traffic. Congestion can quickly spread out and transmit back-



(a)


traffic direction

0:46:00	52.8	49.8	46.4	42.4	37.3	31.7	26.4	21.4	16.5	12.8	5.9
0:46:20	52.5	49.9	46.4	42.0	36.4	31.3	26.8	22.1	17.3	12.9	6.1
0:46:40	52.5	49.8	46.4	41.7	35.8	30.6	26.7	22.9	18.0	12.3	5.8
0:47:00	52.4	49.7	46.2	41.6	35.9	31.0	26.2	22.0	17.9	11.9	3.7
0:47:20	52.3	49.8	46.0	41.3	36.2	31.4	26.8	21.9	17.0	12.1	4.5
0:47:40	52.2	49.5	45.8	41.0	36.3	31.5	26.6	22.1	17.4	12.5	4.6
0:48:00	52.1	49.3	45.9	41.1	36.4	31.4	26.4	21.7	16.8	11.1	3.9
0:48:20	52.1	49.0	45.6	41.3	36.4	31.4	26.0	21.0	16.4	10.8	3.3
0:48:40	51.9	49.1	45.3	41.3	36.4	31.3	25.9	20.8	16.1	11.3	4.9
0:49:00	51.8	48.9	45.1	41.0	36.2	30.5	25.2	20.7	16.2	12.4	6.0
0:49:20	51.7	48.9	45.1	40.9	35.9	30.3	25.4	21.2	16.3	11.0	6.1
0:49:40	51.3	48.7	45.2	40.6	35.5	30.3	25.6	21.6	16.8	11.7	5.7
0:50:00	51.3	48.6	45.1	40.5	35.2	30.4	26.1	22.1	17.7	13.2	6.0
0:50:20	51.1	48.4	44.8	40.3	35.3	30.5	26.0	21.5	17.1	12.5	6.8
0:50:40	51.2	48.1	44.7	40.2	35.2	30.6	25.9	21.1	16.3	11.5	5.6
0:51:00	51.3	48.1	44.4	40.0	35.4	30.9	26.6	21.8	16.7	11.6	4.1
0:51:20	51.3	48.2	44.5	40.0	35.3	31.0	26.2	21.5	17.2	11.8	4.2
0:51:40	51.3	48.3	44.6	40.2	35.4	30.6	25.8	21.0	16.7	11.4	4.7
0:52:00	51.1	48.3	44.6	40.3	35.8	30.8	25.7	21.3	16.5	10.9	4.9
0:52:20	51.1	48.5	44.5	40.1	35.7	30.6	25.6	21.2	16.1	10.9	5.7
0:52:40	51.0	48.5	44.5	39.9	35.4	30.3	25.7	21.4	16.5	10.6	3.1
0:53:00	50.8	48.2	44.5	39.8	35.2	30.4	26.1	21.7	16.8	10.9	3.3
0:53:20	50.7	48.0	44.3	39.8	35.2	30.5	25.9	21.5	16.8	16.1	26.6
0:53:40	50.9	47.8	44.1	39.8	35.3	30.3	25.8	22.5	21.5	27.9	44.6
0:54:00	50.9	47.8	44.3	40.0	35.4	31.1	27.7	26.9	30.0	39.2	52.8
0:54:20	50.9	47.8	44.4	40.3	36.2	32.4	30.7	32.5	37.5	45.9	56.8
0:54:40	50.9	47.8	44.6	41.0	37.3	35.4	35.2	36.5	41.9	49.5	58.0
0:55:00	50.8	48.0	44.8	41.7	39.0	37.8	38.6	41.3	45.0	50.5	57.4
0:55:20	50.8	48.2	45.4	42.5	40.8	40.3	41.1	43.1	47.4	52.1	56.1
0:55:40	51.0	48.5	45.9	43.7	42.4	42.2	43.0	45.3	48.1	51.9	56.4
0:56:00	51.1	48.8	46.7	44.8	43.8	43.8	44.5	46.9	49.7	51.9	55.0
0:56:20	51.2	49.3	47.3	46.1	45.3	45.2	46.2	47.9	50.3	52.4	54.4
0:56:40	51.2	49.7	48.1	46.9	46.6	46.7	47.4	49.2	51.0	52.0	53.5
0:57:00	51.4	49.9	48.8	47.9	47.5	47.9	49.1	50.5	51.4	52.5	53.4
0:57:20	51.7	50.5	49.4	48.7	48.4	49.0	50.3	51.2	51.8	52.3	52.9
0:57:40	52.1	51.1	50.3	49.6	49.6	49.9	50.5	51.3	51.8	51.7	52.3
0:58:00	52.4	51.6	50.9	50.6	50.4	50.7	51.2	51.4	51.5	52.0	52.4
0:58:20	52.7	52.2	51.6	51.3	51.2	51.3	51.3	51.5	51.8	51.4	52.2
0:58:40	53.0	52.6	52.3	51.9	51.8	52.0	51.7	51.4	51.4	51.9	51.6
0:59:00	53.4	52.9	52.8	52.4	52.4	52.3	52.0	51.7	52.0	51.6	51.5
0:59:20	53.7	53.4	53.1	52.9	52.8	52.7	52.3	51.9	51.3	51.2	51.2
0:59:40	53.9	53.8	53.5	53.2	53.2	52.9	52.5	52.1	51.7	50.9	51.1

(b)

Figure 4.9: The simulation results of testing the fundamental diagram and traffic dynamic characteristics in the CAV-enabled traffic using our model are presented.

ward due to the timely communication among vehicles at much stronger magnitudes. These unique traffic equilibrium features and dynamic characteristics are different from these of traditional traffic flow. Further research efforts are needed to fully investigate their theoretical formulation based on the developed simulation models.

In the following experiments, we examine and visualize the variation of the density and velocity distribution under three scenarios individually. In the first scenario, all the forces are turned off except the pressure force $\mathbf{f}_p$. The traffic density distribution is directly specified by the user to be highly nonuniform. Eventually, the distribution spreads out smoothly as shown in Fig. 4.10 (a). Similarly in the second scenario, only the viscosity force $\mathbf{f}_v$ is turned on. One particle is activated and has a constant

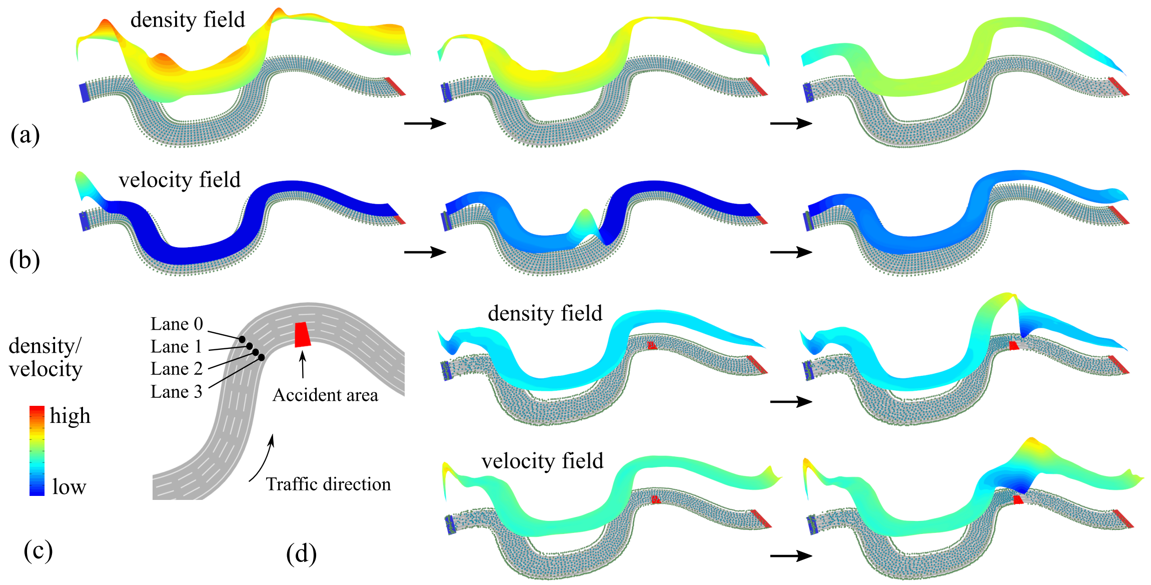


Figure 4.10: The simulation results show how does the CAV-enabled traffic response to the imposed density change (a), velocity change (b), and under the traffic congestion (d). The color map (c) is associated with the density and velocity field, where red represents the highest value and blue the lowest value. Please refer to the accompanying video for details.

prescribed velocity. All other particles are stationary initially. As expected, the activated particle finally drives all particles to a certain velocity level as shown in Fig. 4.10 (b). In the third scenario, the traffic flow in a four-lane roadway is initially in an equilibrium state. An artificial crash event is created and two lanes (Lane 2 and 3) are blocked (shown in Fig. 4.10 (d)). The velocity within the accident area is constrained. We can see that the congestion transmits along both the traffic direction and the normal direction. The congestion can be partially mitigated through the available capacity from the adjacent lanes. This means that the proposed model not only can simulate the traffic flow along the traffic direction as existing traffic models do, but also can automatically handle the lane-changing behavior of the traffic flow (i.e. vehicles moving along the normal direction), because of its two-dimensional nature.

In order to better understand the CAV traffic flow characteristics, further experimental tests are conducted to quantify the dynamic performance of the developed model. The simulation results are shown in Fig. 4.11. We can see that although there is no accident in Lane 0, congestion formed and propagated. Note that we don't have an explicit lane-changing mechanism in the simulation, our model automatically handles the lane-changing requirement in this scenario. We also observe an interesting "speed up" area as highlighted in the figure. That is because the density in the upstream is high which increases the pressure force to push the downstream.

4.6 Conclusions

In this study, a new force-driven continuum traffic flow model is proposed and SPH-based numerical solutions are provided. The proposed model amplifies neighborhood-wide vehicle coupling characterized in CAV-enabled traffic systems, and individual vehicles spontaneously seek for local optimal configurations based on the real-time information shared/obtained from the surroundings. Inspired by the similarity between natural physical particle flow and CAV traffic flow, we formulate and simulate the mutually-coupled vehicle interactions using internal virtual forces. Our experiment shows that the proposed model can be downgraded to incorporate the existing macroscopic traffic flow model and accurately describes the real-world traffic. Thus, there is a reason to believe this model works well in the context of CAV-enabled traffic systems too. Extensive experiments are conducted to illustrate the new fundamental diagram under the stationary equilibrium conditions and dynamic congestion formation/dissipation in the CAV-enabled traffic. Various operational scenarios are designed and tested to further demonstrate CAV traffic flow characteristics.

The developed CAV traffic flow model can advance our understanding of CAV-enabled traffic flow operations, facilitate near-future CAV-penetrated traffic management, quantify existing traffic facility capacity in the context of CAVs, and optimize

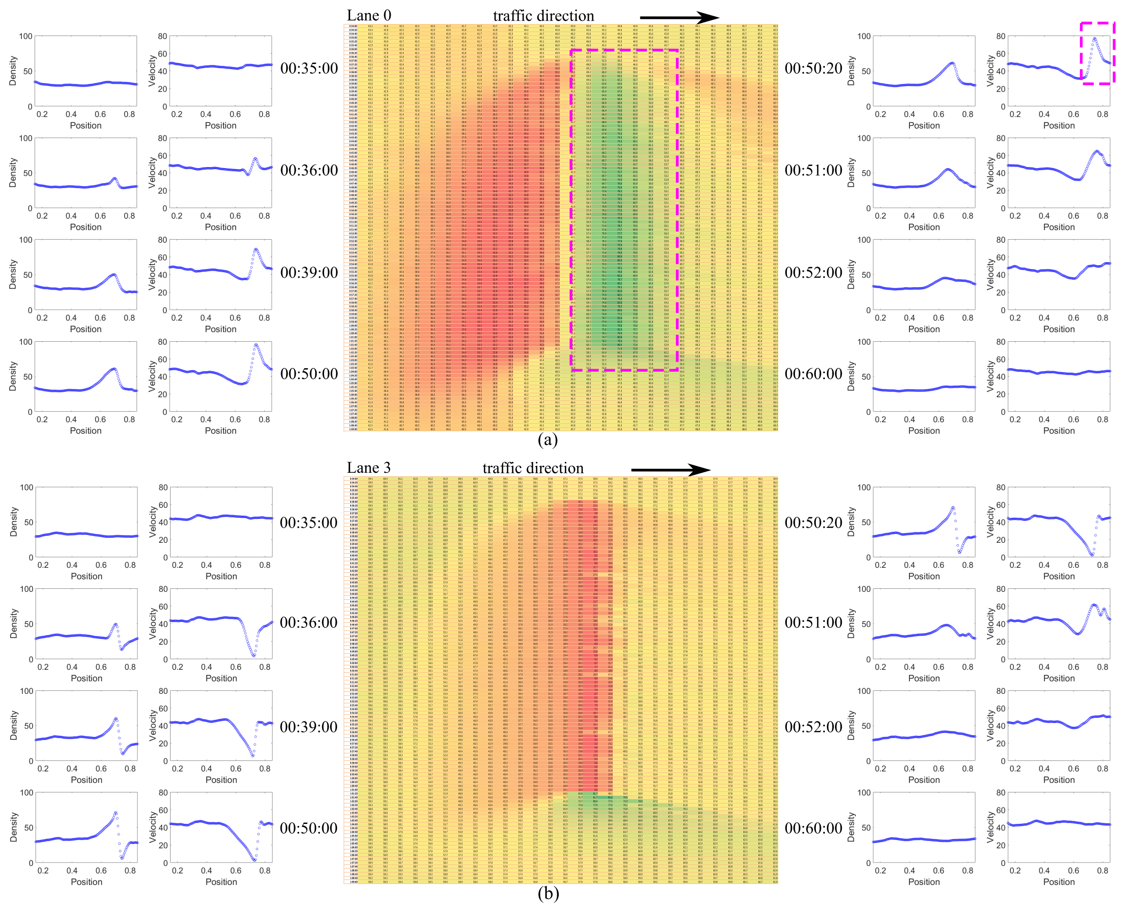


Figure 4.11: The results of the congestion simulation in the CAV-enabled traffic. An artificial accident occurs at Lane 2 and 3 and the velocity within the accident area is constrained to zero. After 15 minutes, the accident is cleared. We can see that the congestion transmits not only horizontally (backward in Lane 2 and 3) but also vertically (to the adjacent Lane 0 and 1 where no accidents occurred though) due to the two-dimensional nature of our model. An interesting “speed-up” area in Lane 0 is also highlighted.

mixed-traffic-to-infrastructure interactions. For example, based on the developed model, various traffic operation management and control strategies can be tested theoretically and practically to verify their effectiveness before their implementations in CAV-enabled traffic flow. The latest share mobility modeling can be enhanced and smart decision making can be supported further in the interconnected and automated traffic environment. The theoretical insights extracted from the developed model can greatly facilitate CAV deployment in the entire traffic operation research domain.

Chapter 5

Discussion

5.1 Summary

We investigate and address a couple of transportation problems with respect to geographic discretization, autonomous pavement surface reconstruction and examination, and future CAV-enable traffic flow simulation, using advanced computational technologies. We developed an efficient discretization method that generates a list of nearby road segments associated to individual cells. More than that, the method is extended to 3D scenarios to voxelize complex triangle meshes. The proposed method outperforms the conventional methods especially under the high-resolution discretization requirements. Based on Kinect technologies, we developed a cost-effective pavement surface assessment solution which integrates 3D pavement surface reconstruction and autonomous crack analysis. Experiments show that the solution provides promising measurement results with a really low system cost compared to the existing counterparts. To simulate the future CAV-enable traffic flow, we proposed a force-driven macroscopic traffic flow model based on the mutually-coupled characteristics of CAVs. When downgraded to incorporate the existing conventional model, the proposed model is validated by the real-world traffic data and the simulation result shows great consistency. This gives us a reason to believe that the model works well in the context of future CAV-enabled traffic systems.

5.2 Future Work

The research in this dissertation leaves us many interesting further directions to explore. For the efficient discretization, the current version of our method is not able to produce the voxelization with adaptive resolutions (e.g. like using an octree). This limitation may be resolved by applying our method at various density levels incrementally. It is also worthy to further investigate how to utilize the superior performance of the proposed method – we see runtime collision culling is a promising application.

Currently our pavement surface recognition and cracking analysis system is purely based on the reconstructed 3D triangle mesh. We could potentially improve the measurement accuracy by combining with the 2D RGB surface images. Although Microsoft officially announced the discontinuation of Kinect on 10/25/2017, its cost-effective solution for generating depth images is still attractive in numerous research fields, such as robotics, motion tracking, physical rehabilitation, besides 3D reconstruction.

The proposed traffic flow model should be tested by real CAV-enabled traffic data when it is sufficiently available. So far, the proposed virtual forces can only model the interaction of the CAVs in the same simulation domain or with the same traffic direction, and the information brought by the CAVs in the opposite direction is ignored. In order to handle the interaction between CAVs with opposite directions, we need to develop new types of virtual forces. Similarly, the proposed model is not sufficient for modeling the traffic with CAVs and HDVs mixed traffic behavior.

This kind of interaction is basically asymmetric and might be modeled by some asymmetric virtual forces.

Note the virtual-force-based parameters are manually tuned, it would be an interesting topic that how we can formulate those parameters to the real-world traffic flow. We currently use full-kernel and half-kernel support in the CAV-enabled traffic and the conventional traffic respectively based on their different microscopic behavior. It is not clear yet how does the kernel function mathematically affect the fundamental diagram. It is also interesting to use grid-based simulation [Bri15] instead of the particle-based one to simulate the proposed model. In this case, the convection (as appeared in Equation (4.5)) must be incorporated. Additionally, our current simulation is running only on CPU. We will further port our algorithm to GPU to fully leverage the parallelism of the SPH method. We will put more efforts on the related research in the future.

Appendix A

Derivation of Eq. 2.7

Let L_1 be the line segment connecting $\mathbf{P}_A(X_A, Y_A)$ and $\mathbf{P}_B(X_B, Y_B)$ and $\Delta X_{AB} = X_A - X_B$, $\Delta Y_{AB} = Y_A - Y_B$ as defined in Eq. 2.6. The line equation of L_1 can be written in the form of $ax + by + c = 0$, where:

$$a = \Delta Y_{AB}, \qquad b = \Delta X_{AB}, \qquad c = \Delta X_{AB} Y_A - \Delta Y_{AB} X_A.$$

It is known that the distance between a point (x_0, y_0) and $ax + by + c = 0$ is $|ax_0 + by_0 + c|/\sqrt{a^2 + b^2}$, which leads to:

$$d_A = \frac{1}{\sqrt{\Delta X_{AB}^2 + \Delta Y_{AB}^2}} \Big| \Delta Y_{AB}(X'_A - X_A) - \Delta X_{AB}(Y'_A - Y_A) \Big|.$$

References

[Adh06] Sondipon Adhikari. Damping modelling using generalized proportional damping. *Journal of Sound and Vibration*, 293(1):156–170, 2006.

[ADR16] Boris Andreianov, Carlotta Donadello, and Massimiliano Daniele Rosini. A second-order model for vehicular traffics with local point constraints on the flow. *Mathematical Models and Methods in Applied Sciences*, 26(04):751–802, 2016.

[AKT16] Gowri Asaithambi, Venkatesan Kanagaraj, and Tomer Toledo. Driving behaviors: Models and challenges for non-lane based mixed traffic. *Transportation in Developing Economies*, 2(2):19, 2016.

[AM05] Tomas Akenine-Möller. Fast 3D triangle-box overlap testing. In *ACM SIGGRAPH 2005 Courses*, page 8. ACM, 2005.

[AMA05] Tomas Akenine-Möller and Timo Aila. Conservative and tiled rasterization using a modified triangle set-up. *Journal of Graphics, GPU, and Game Tools*, 10(3):1–8, 2005.

[AMHH08] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-time rendering*. CRC Press, 2008.

[AW$^+$87] John Amanatides, Andrew Woo, et al. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87, page 10, 1987.

[AW11] Chikit Au and Tony Woo. Three dimensional extension of bresenham's algorithm with voronoi diagram. *Computer-Aided Design*, 43(4):417–426, 2011.

[BB02] Valentin E Brimkov and Reneta P Barneva. Graceful planes and lines. *Theoretical Computer Science*, 283(1):151–170, 2002.

[BBH$^+$01] Liviu Bursanescu, Mihaela Bursanescu, Maher Hamdi, Alain Lardigue, and Damien Paiement. Three-dimensional infrared laser vision system for road surface features analysis. In *ROMOPTO 2000: Sixth Conference on Optics*, pages 801–808. International Society for Optics and Photonics, 2001.

[BEG11] J Bender, K Erleben, and E Galin. Sph based shallow water simulation. 2011.

[Ber97] Gino van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, 2(4):1–13, 1997.

[BHM$^+$12] Paola Bandini, Susan Bogus Halter, Kelly R Montoya, Hung V Pham, and Giovanni C Migliaccio. Improving nmdot's pavement distress survey methodology and developing correlations between fhwa's hpms distress data and pms data. Technical report, 2012.

[BM92] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[BMC10] Susan Bogus, Giovanni Migliaccio, and Arturo Cordova. Assessment of data quality for evaluations of manual pavement distress. *Transportation Research Record: Journal of the Transportation Research Board*, (2170):1–8, 2010.

[Bre65] Jack E Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems journal*, 4(1):25–30, 1965.

[Bri15] Robert Bridson. *Fluid simulation for computer graphics.* CRC Press, 2015.

[BVLH06] Justin Bray, Brijesh Verma, Xue Li, and Wade He. A neural network based technique for automatic classification of road cracks. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 907–912. IEEE, 2006.

[BW01] David E. Breen and Ross T. Whitaker. A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):173–192, 2001.

[CG12] Cyril Crassin and Simon Green. Octree-based sparse voxelization using the gpu hardware rasterizer. *OpenGL Insights*, pages 303–318, 2012.

[CGG+16] Sandro Chiappone, Orazio Giuffrè, Anna Granà, Raffaele Mauro, and Antonino Sferlazza. Traffic simulation models calibration using speed–density relationship: an automated procedure based on genetic algorithm. *Expert Systems with Applications*, 44:147–155, 2016.

[CGMN10] Sylvie Chambon, Christian Gourraud, Jean Marc Moliard, and Philippe Nicolle. Road crack extraction with adapted filtering and markov model-based segmentation: introduction and validation. In *International Joint Conference on Computer Vision Theory and Applications, VISAPP*, page sp, 2010.

[CGW17] Rongjun Cheng, Hongxia Ge, and Jufeng Wang. An extended continuum model accounting for the driver's timid and aggressive attributions. *Physics Letters A*, 381(15):1302–1312, 2017.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[CLB+09] Ming Chuang, Linjie Luo, Benedict J Brown, Szymon Rusinkiewicz, and Michael Kazhdan. Estimating the laplace-beltrami operator by restricting 3d functions. In *Computer Graphics Forum*, volume 28, pages 1475–1484. Wiley Online Library, 2009.

[CLZ+12] Chien-Yen Chang, Belinda Lange, Mi Zhang, Sebastian Koenig, Phil Requejo, Noom Somboon, Alexander A Sawchuk, and Albert A Rizzo. Towards pervasive physical rehabilitation using microsoft kinect. In *2012 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pages 159–162. IEEE, 2012.

[COK95] Daniel Cohen-Or and Arie Kaufman. Fundamentals of surface voxelization. *Graphical models and image processing*, 57(6):453–461, 1995.

[COK97] Daniel Cohen-Or and Arie Kaufman. 3D line voxelization and connectivity control. *Computer Graphics and Applications, IEEE*, 17(6):80–87, 1997.

[CR99] Sharen J Cummins and Murray Rudman. An sph projection method. *Journal of computational physics*, 152(2):584–607, 1999.

[CWDY15] Yongjing Chao, Yingmei Wei, Xiaolei Du, and Fang Yuan. Alarm thresholds of threaten regions based on triangle mesh voxelization. In

Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on, pages 2475–2479. IEEE, 2015.

[CZZ$^+$14] Cong Chen, Su Zhang, Guohui Zhang, Susan M Bogus, and Vanessa Valentin. Discovering temporal and spatial patterns and characteristics of pavement distress condition data on major corridors in new mexico. *Journal of Transport Geography*, 38:148–158, 2014.

[D$^+$14] Jerome Daleiden et al. Variability of pavement distress data from manual surveys. In *Pavement Evaluation Conference 2014*, 2014.

[DB08] Jie Du and Matthew J Barth. Next-generation automated vehicle location systems: Positioning at the lane level. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):48–57, 2008.

[DCB$^+$04] Zhao Dong, Wei Chen, Hujun Bao, Hongxin Zhang, and Qunsheng Peng. Real-time voxelization for complex polygonal models. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on*, pages 43–50. IEEE, 2004.

[DD15] Lili Du and Hoang Dao. Information dissemination delay in vehicle-to-vehicle communication networks in a traffic stream. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):66–80, 2015.

[DKB$^+$16] Bas Dado, Timothy R Kol, Pablo Bauszat, Jean-Marc Thiery, and Elmar Eisemann. Geometry and attribute compression for voxel scenes. In *Computer Graphics Forum*, volume 35, pages 397–407. Wiley Online Library, 2016.

[DMSB00] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators in nd. *preprint, the Caltech Multi-Res Modeling Group*, 2000.

[Ebe01] David Eberly. Intersection of convex objects: The method of separating axes. *www. magic-software. com*, 2001.

[ED08] Elmar Eisemann and Xavier Décoret. Single-pass gpu solid voxelization for real-time applications. In *Proceedings of graphics interface 2008*, pages 73–80. Canadian Information Processing Society, 2008.

[FAMO99] Ronald P Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of computational physics*, 152(2):457–492, 1999.

[Fed02] Ronald P Fedkiw. Coupling an eulerian fluid calculation to a lagrangian solid calculation with the ghost fluid method. *Journal of Computational Physics*, 175(1):200–224, 2002.

[FHKZ15] Yiheng Feng, K Larry Head, Shayan Khoshmagham, and Mehdi Zamanipour. A real-time adaptive signal control in a connected vehicle environment. *Transportation Research Part C: Emerging Technologies*, 55:460–473, 2015.

[FM09] Gerardo W Flintsch and Kevin K McGhee. *Quality management of pavement condition data collection*, volume 401. Transportation Research Board, 2009.

[FWC12] Yun Fei, Bin Wang, and Jiating Chen. Point-tessellated voxelization. In *Proceedings of Graphics Interface 2012*, pages 9–18. Canadian Information Processing Society, 2012.

[GH93] Wade L Gramling and John E Hunt. Photographic pavement distress record collection and transverse profile analysis. Technical report, Strategic Highway Research Program, National Research Council, 1993.

[GHP59] Denos C Gazis, Robert Herman, and Renfrey B Potts. Car-following theory of steady-state traffic flow. *Operations research*, 7(4):499–505, 1959.

[GLF95] A Georgopoulos, A Loizos, and A Flouda. Digital image processing as a tool for pavement distress evaluation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 50(1):23–33, 1995.

[GLM96] Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180. ACM, 1996.

[GM77] Robert A Gingold and Joseph J Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

[GP17] Mauro Garavello and Benedetto Piccoli. Boundary coupling of microscopic and first order macroscopic traffic models. *Nonlinear Differential Equations and Applications NoDEA*, 24(4):43, 2017.

[Gra] W Gramling. Nchrp synthesis of highway practice 203: Current practices in determining pavement condition. trb, national research council, washington, d. c., 1994.

[GZY$^+$17] Lantian Guo, Xiangmo Zhao, Shaowei Yu, Xiuhai Li, and Zhongke Shi. An improved car-following model with multiple preceding cars velocity fluctuation feedback. *Physica A: Statistical Mechanics and its Applications*, 471:436–444, 2017.

[HAMO05] Jon Hasselgren, Tomas Akenine-Möller, and Lennart Ohlsson. Conservative rasterization. *GPU Gems*, 2:677–690, 2005.

[HCTS10] Hsien-Hsi Hsieh, Chin-Chen Chang, Wen-Kai Tai, and Han-Wei Shen. Novel geometrical voxelization approach with application to streamlines. *Journal of Computer Science and Technology*, 25(5):895–904, 2010.

[HH09] Jiaxi Hu and Jing Hua. Salient spectral geometric features for shape matching and retrieval. *The visual computer*, 25(5-7):667–675, 2009.

[HHZ94] Ralph Haas, W Ronald Hudson, and John P Zaniewski. *Modern pavement management*. 1994.

[HK97] Taosong He and Arie Kaufman. Collision detection for volumetric objects. In *Proceedings of the 8th conference on Visualization'97*, pages 27–ff. IEEE Computer Society Press, 1997.

[HPL15] Jia Hu, Byungkyu Brian Park, and Young-Jae Lee. Coordinated transit signal priority supporting transit progression under connected vehicle technology. *Transportation Research Part C: Emerging Technologies*, 55:393–408, 2015.

[HR17] Helge Holden and Nils Henrik Risebro. Continuum limit of follow-the-leader models. *arXiv preprint arXiv:1702.01718*, 2017.

[Hsu11] Hui-mei Justina Hsu. The potential of kinect in education. *International Journal of Information and Education Technology*, 1(5):365, 2011.

[HTB03] Dirk Haehnel, Sebastian Thrun, and Wolfram Burgard. An extension of the icp algorithm for modeling nonrigid objects with mobile robots. In *IJCAI*, volume 3, pages 915–920, 2003.

[HW94] Eric A Haines and John R Wallace. Shaft culling for efficient ray-cast radiosity. In *Photorealistic Rendering in Computer Graphics*, pages 122–138. Springer, 1994.

[HX06] Yaxiong Huang and Bugao Xu. Automatic inspection of pavement cracking distress. *Journal of Electronic Imaging*, 15(1):013017–013017, 2006.

[HYFK98] Jian Huang, Roni Yagel, Vassily Filippov, and Yair Kurzion. An accurate method for voxelizing polygon meshes. In *Volume Visualization, IEEE Symposium on*, pages 119–126. IEEE, 1998.

[IDC09] Paulo Ivson, Leonardo Duarte, and Waldemar Celes. Gpu-accelerated uniform grid construction for ray tracing dynamic scenes. *Master's thesis, Departamento de Informatica, Pontificia Universidade Catolica, Rio de Janeiro*, 2009.

[IKH$^+$11] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[JBS06] Mark W Jones, J Andreas Baerentzen, and Milos Sramek. 3d distance fields: A survey of techniques and applications. *IEEE Transactions on visualization and Computer Graphics*, 12(4):581–599, 2006.

[JBT04] Doug L. James, Jernej Barbič, and Christopher D. Twigg. Squashing cubes: Automating deformable model construction for graphics. In *ACM SIGGRAPH 2004 Sketches*, SIGGRAPH '04, page 38, 2004.

[JJMBG12] Mohammad R Jahanshahi, Farrokh Jazizadeh, Sami F Masri, and Burcin Becerik-Gerber. Unsupervised approach for autonomous pavement-defect detection and quantification using an inexpensive depth sensor. *Journal of Computing in Civil Engineering*, 27(6):743–754, 2012.

[JO14] I Ge Jin and Gábor Orosz. Dynamics of connected vehicle systems with delayed acceleration feedback. *Transportation Research Part C: Emerging Technologies*, 46:46–64, 2014.

[JWZ$^+$13] Peter J Jin, C Michael Walton, Guohui Zhang, Xiaowen Jiang, and Amit Singh. Analyzing the impact of false-accident cyber attacks on traffic flow stability in connected vehicle environment. In *Connected Vehicles and Expo (ICCVE), 2013 International Conference on*, pages 616–621. IEEE, 2013.

[KAMM13] Naofumi Kitsunezaki, Eijiro Adachi, Takashi Masuda, and Jun-ichi Mizusawa. Kinect applications for the physical rehabilitation. In *Medical Measurements and Applications Proceedings (MeMeA), 2013 IEEE International Symposium on*, pages 294–299. IEEE, 2013.

[Kau87] Arie Kaufman. Efficient algorithms for 3d scan-conversion of parametric curves, surfaces, and volumes. *ACM SIGGRAPH Computer Graphics*, 21(4):171–179, 1987.

[Kau88] Arie Kaufman. Efficient algorithms for scan-converting 3d polygons. *Computers & Graphics*, 12(2):213–219, 1988.

[KB11] Christian Koch and Ioannis Brilakis. Pothole detection in asphalt pavement images. *Advanced Engineering Informatics*, 25(3):507–515, 2011.

[KE12] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[KFR03] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164, 2003.

[KIH$^+$15] Md Abdus Samad Kamal, Jun-ichi Imura, Tomohisa Hayakawa, Akira Ohata, and Kazuyuki Aihara. A vehicle-intersection coordination scheme for smooth flows of traffic without using traffic lights. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1136–1147, 2015.

[KJ$^+$01] Andreas Kolb, Lars John, et al. Volumetric model repair for virtual reality applications. *EG, Short-Paper*, pages 249–256, 2001.

[KK99] Kevin Kreeger and Arie Kaufman. Mixing translucent polygons with volumes. In *Proceedings of the conference on Visualization'99*, pages 191–198, 1999.

[KLG11] Agop Koulakezian and Alberto Leon-Garcia. Cvi: Connected vehicle infrastructure for its. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium on*, pages 750–755. IEEE, 2011.

[KS87] Arie Kaufman and Eyal Shimony. 3d scan-conversion algorithms for voxel-based graphics. In *Proceedings of the 1986 workshop on Interactive 3D graphics*, pages 45–75. ACM, 1987.

[KTY16] Md Abdus Samad Kamal, Shun Taguchi, and Takayoshi Yoshimura. Efficient driving on multilane roads under a connected vehicle environment. *IEEE Transactions on Intelligent Transportation Systems*, 17(9):2541–2551, 2016.

[KYL$^+$11] Farid Abedan Kondori, Shahrouz Yousefi, Haibo Li, Samuel Sonning, and Sabina Sonning. 3d head pose estimation using the kinect. In *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*, pages 1–4. IEEE, 2011.

[LC87] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[LC02] XW Liu and K Cheng. Three-dimensional extension of bresenham's algorithm and its application in straight-line interpolation. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 216(3):459–463, 2002.

[LFWK05] Wei Li, Zhe Fan, Xiaoming Wei, and Arie Kaufman. Flow simulation with complex boundaries. *GPU Gems*, 2:747–764, 2005.

[LG11] Alberto Leon-Garcia. Smart infrastructure and applications for connected vehicles. ITS World Congress,Orlando, Fl, 2011.

[LGG01] Christian Langis, Michael Greenspan, and Guy Godin. The parallel iterative closest point algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 195–202. IEEE, 2001.

[LGJA$^+$] Alberto Leon-Garcia, Hans-Arno Jacobsen, Baher Adbulhai, Marin Litoiu, and Ali Tizghadam. Connected vehicles and smart transportation (cvst) portal. *https://www.utoronto.ca/news/challenge-u-t-engineering-team-one-eight-selected-develop-self-driving-electric-cars.*

[LGK11] Alberto Leon-Garcia and Agop Koulakezian. Participatory sensing for its. ITS Canada AGCM, Vancouver, 2011.

[LHLW10] Fang Liu, Meng-Cheng Huang, Xue-Hui Liu, and En-Hua Wu. Freepipe: a programmable parallel rendering architecture for efficient multi-fragment effects. In *I3D*, pages 75–82, 2010.

[Lit14] Todd Litman. Autonomous vehicle implementation predictions. *Victoria Transport Policy Institute*, 28, 2014.

[LKM$^+$12] Belinda Lange, Sebastian Koenig, Eric McConnell, Chien-Yen Chang, Rick Juang, Evan Suma, Mark Bolas, and Albert Rizzo. Interactive game-based rehabilitation using the microsoft kinect. In *2012 IEEE Virtual Reality Workshops (VRW)*, pages 171–172. IEEE, 2012.

[LL10] MB Liu and GR Liu. Smoothed particle hydrodynamics (sph): an overview and recent developments. *Archives of computational methods in engineering*, 17(1):25–76, 2010.

[LLC16] Albert YS Lam, Yiu-Wing Leung, and Xiaowen Chu. Autonomous-vehicle public transportation system: scheduling and admission control. *IEEE Transactions on Intelligent Transportation Systems*, 17(5):1210–1226, 2016.

[LM00] Zhen Luo and Margaret Martonosi. Accelerating pipelined integer and floating-point accumulations in configurable hardware with delayed addition techniques. *Computers, IEEE Transactions on*, 49(3):208–218, 2000.

[LMG02] Jian John Lu, Xiaoyu Mei, and Manjriker Gunaratne. Development of an automatic detection system for measuring pavement crack depth on florida roadways. 2002.

[LP12] Joyoung Lee and Byungkyu Park. Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):81–90, 2012.

[LPG12] Yang Liu, Balakrishnan Prabhakaran, and Xiaohu Guo. Point-based manifold harmonics. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1693–1703, 2012.

[LPMS13] Joyoung Lee, Byungkyu Brian Park, Kristin Malakorn, and Jaehyun Jason So. Sustainability assessments of cooperative vehicle intersection control at an urban corridor. *Transportation Research Part C: Emerging Technologies*, 32:193–206, 2013.

[LS80] Der-Tsai Lee and Bruce J Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.

[LTD97] John Laurent, Mario Talbot, and Michel Doucet. Road surface inspection using laser scanners adapted for the high precision 3d measurements of large flat surfaces. In *3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in*, pages 303–310. IEEE, 1997.

[LTTA11] Panagiotis Lytrivis, George Thomaidis, Manolis Tsogas, and Angelos Amditis. An advanced cooperative path prediction algorithm for safety applications in vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):669–679, 2011.

[Luc77] Leon B Lucy. A numerical approach to the testing of the fission hypothesis. *The astronomical journal*, 82:1013–1024, 1977.

[LW55] Michael J Lighthill and Gerald Beresford Whitham. On kinematic waves. ii. a theory of traffic flow on long crowded roads. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 229, pages 317–345. The Royal Society, 1955.

[LXXQ17] Zhipeng Li, Xun Xu, Shangzhi Xu, and Yeqing Qian. A heterogeneous traffic flow model consisting of two types of vehicles with different sensitivities. *Communications in Nonlinear Science and Numerical Simulation*, 42:132–145, 2017.

[LZK$^+$13] Xiaoyue Liu, Guohui Zhang, Carmen Kwan, Yinhai Wang, and Brian Kemper. Simulation-based, scenario-driven integrated corridor management strategy analysis. *Transportation Research Record: Journal of the Transportation Research Board*, (2396):38–44, 2013.

[LZZ$^+$09] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching for low-sampling-rate gps trajectories. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 352–361. ACM, 2009.

[MBL84] Panos G Michalopoulos, Dimitrios E Beskos, and Jaw-Kuan Lin. Analysis of interrupted traffic flow by finite difference methods. *Transportation Research Part B: Methodological*, 18(4):409–421, 1984.

[MBSEF14] Julien Monteil, Romain Billot, Jacques Sau, and Nour-Eddin El Faouzi. Linear and weakly nonlinear stability analyses of cooperative car-following models. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2001–2013, 2014.

[MCG03] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159. Eurographics Association, 2003.

[McG04] Kenneth H McGhee. *Automated pavement distress collection techniques*, volume 334. Transportation Research Board, 2004.

[MKWS91] David S Mahler, Zuhair B Kharoufa, Edward K Wong, and Leonard G Shaw. Pavement distress analysis using image processing techniques. *Computer-Aided Civil and Infrastructure Engineering*, 6(1):1–14, 1991.

[MLP08] M Mustaffara, TC Lingb, and OC Puanb. Automated pavement imaging program (apip) for pavement cracks classification and quantification-a photogrammetric approach. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37(B4):367–372, 2008.

[Mon92] Joe J Monaghan. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics*, 30(1):543–574, 1992.

[MSV95] Ravi Malladi, James A Sethian, and Baba C Vemuri. Shape modeling with front propagation: A level set approach. *IEEE transactions on pattern analysis and machine intelligence*, 17(2):158–175, 1995.

[MWM01] Michael D McCool, Chris Wales, and Kevin Moule. Incremental and hierarchical hilbert order edge equation polygon rasterizatione. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 65–72. ACM, 2001.

[MZH08] Chang-Xia Ma, Chun-Xia Zhao, and Ying-kun Hou. Pavement distress detection based on nonsubsampled contourlet transform. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 1, pages 28–31. IEEE, 2008.

[NDS10] Pavol Novotny, Leonid I Dimitrov, and Milos Sramek. Enhanced voxelization and representation of objects with sharp details in truncated distance fields. *IEEE transactions on visualization and computer graphics*, 16(3):484–498, 2010.

[NIH$^+$11] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[NLAW11] Daiheng Ni, Jia Li, Steven Andrews, and Haizhong Wang. A methodology to estimate capacity impact due to connected vehicle technology. *International Journal of Vehicular Technology*, 2012, 2011.

[Nob10] Tsuguo Nobe. Connected vehicle accelerates green driving. *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, 3(2010-01-2315):68–75, 2010.

[NSSL13] Matthias Nießner, Christian Siegl, Henry Schäfer, and Charles Loop. Real-time collision detection for dynamic hardware tessellated objects. 2013.

[OC08] Henrique Oliveira and Paulo Lobato Correia. Identifying and retrieving distress images from road pavement surveys. In *2008 15th IEEE International Conference on Image Processing*, pages 57–60. IEEE, 2008.

[Oh98] Hyungkee Oh. Image processing technique in automated pavement evaluation system. 1998.

[OI15] Osama A Osman and Sherif Ishak. A network level connectivity robustness measure for connected vehicle environments. *Transportation Research Part C: Emerging Technologies*, 53:48–58, 2015.

[OKWM12] Ayrton Oliver, Steven Kang, Burkhard C Wünsche, and Bruce MacDonald. Using the kinect as a navigation sensor for mobile robotics. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*, pages 509–514. ACM, 2012.

[Pan11] Jacopo Pantaleoni. Voxelpipe: a programmable pipeline for 3d voxelization. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, pages 99–106. ACM, 2011.

[Par08] Hyungjun Park. *Development of ramp metering algorithms using individual vehicular data and control under vehicle infrastructure integration*, volume 70. 2008.

[PBHS89] Markos Papageorgiou, Jean-Marc Blosseville, and Habib Hadj-Salem. Macroscopic modelling of traffic flow on the boulevard périphérique in paris. *Transportation Research Part B: Methodological*, 23(1):29–47, 1989.

[PF01] Ronald N Perry and Sarah F Frisken. Kizamu: A system for sculpting digital characters. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 47–56. ACM, 2001.

[Pin88] Juan Pineda. A parallel algorithm for polygon rasterization. In *SIGGRAPH*, volume 22, pages 17–20, 1988.

[PWL99] J Pynn, A Wright, and R Lodge. Automatic identification of cracks in road surfaces. In *Image Processing and Its Applications, 1999. Seventh International Conference on (Conf. Publ. No. 465)*, volume 2, pages 671–675. IET, 1999.

[RC02] Hesham Rakha and Brent Crowther. Comparison of greenshields, pipes, and van aerde car-following and traffic stream models. *Transportation Research Record: Journal of the Transportation Research Board*, (1802):248–262, 2002.

[Ric56] Paul I Richards. Shock waves on the highway. *Operations research*, 4(1):42–51, 1956.

[RMY11] Zhou Ren, Jingjing Meng, and Junsong Yuan. Depth camera based hand gesture recognition and its applications in human-computer-interaction. In *Information, Communications and Signal Processing (ICICS) 2011 8th International Conference on*, pages 1–5. IEEE, 2011.

[RNB$^+$07] Marcel Rieser, Kai Nagel, Ulrike Beuck, Michael Balmer, and Jens Rümenapp. Agent-oriented coupling of activity-based demand generation with multiagent traffic simulation. *Transportation Research Record: Journal of the Transportation Research Board*, (2021):10–17, 2007.

[Ros88] Paul Ross. Traffic dynamics. *Transportation Research Part B: Methodological*, 22(6):421–435, 1988.

[SB12] Hagit Schechter and Robert Bridson. Ghost sph for animating water. *ACM Transactions on Graphics (TOG)*, 31(4):61, 2012.

[SBV91] Mark Shand, Patrice Bertin, and Jean Vuillemin. Hardware speedups in long integer multiplication. *ACM SIGARCH Computer Architecture News*, 19(1):106–113, 1991.

[SF95] Jos Stam and Eugene Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 129–136. ACM, 1995.

[SFP96] Roger E Smith, Thomas J Freeman, and Olga J Pendleton. Evaluation of automated pavement distress data collection procedures for local agency pavement management. *Research Project GC10470, Washington State Department of Transportation, Olympia, Washington*, 1996.

[SK98] Milos Sramek and Arie Kaufman. Object voxelization by filtering. In *Volume Visualization, 1998. IEEE Symposium on*, pages 111–118. IEEE, 1998.

[SK99] Milos Sramek and Arie E Kaufman. Alias-free voxelization of geometric objects. *IEEE transactions on visualization and computer graphics*, 5(3):251–267, 1999.

[SLPR13] Andrea Sanna, Fabrizio Lamberti, Gianluca Paravati, and Felipe Domingues Rocha. A kinect-based interface to animate virtual characters. *Journal on Multimodal User Interfaces*, 7(4):269–279, 2013.

[SS10] Michael Schwarz and Hans-Peter Seidel. Fast parallel surface and solid voxelization on gpus. In *ACM Transactions on Graphics (TOG)*, volume 29, page 179. ACM, 2010.

[TA04] Benoit Tremblais and Bertrand Augereau. A fast multi-scale edge detection algorithm. *Pattern Recognition Letters*, 25(6):603–618, 2004.

[Tec13] C. V. Technology. Keeping the promise of connected vehicle technology. pages 3–9, 2013.

[Tem84] Roger Temam. *Navier-stokes equations*, volume 2. North-Holland Amsterdam, 1984.

[TH11] Michal Tölgyessy and Peter Hubinskỳ. The kinect sensor in robotics education. In *Proceedings of 2nd International Conference on Robotics in Education*, pages 143–146, 2011.

[THK16] Mohammed Amine Togou, Abdelhakim Hafid, and Lyes Khoukhi. Scrp: Stable cds-based routing protocol for urban vehicular ad hoc networks. *IEEE Transactions on Intelligent Transportation Systems*, 17(5):1298–1307, 2016.

[THM$^+$03] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H Gross. Optimized spatial hashing for collision detection of deformable objects. In *Vmv*, volume 3, pages 47–54, 2003.

[Tic15] David Ticoll. *Driving changes: Automated vehicles in Toronto.* Munk School of Global Affairs, University of Toronto, 2015.

[TZWZ15] Kamonthep Tiaprasert, Yunlong Zhang, Xiubin Bruce Wang, and Xiaosi Zeng. Queue length estimation using connected vehicle technology for adaptive signal control. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):2129–2140, 2015.

[UKCL10] B Shane Underwood, Y Richard Kim, and J Corley-Lay. Assessment of use of automated distress survey methods for network-level pavement management. *Journal of Performance of Constructed Facilities*, 25(3):250–258, 2010.

[Uof] University of toronto is one of eight universities from across north america chosen to compete in the autodrive challenge, sponsored by gm and sae international. *https://www.utoronto.ca/news/challenge-u-t-engineering-team-one-eight-selected-develop-self-driving-electric-cars.*

[VESS13] William R Vavrik, Lynn D Evans, Joseph A Stefanski, and Shad Sargand. Pcr evaluation–considering transition from manual to semi-automated pavement distress collection and analysis. 2013.

[VKK$^+$03] Gokul Varadhan, Shankar Krishnan, Young J Kim, Suhas Diggavi, and Dinesh Manocha. Efficient max-norm distance computation and reliable voxelization. In *SGP*, pages 116–126, 2003.

[VL08] Bruno Vallet and Bruno Lévy. Spectral geometry processing with manifold harmonics. In *Computer Graphics Forum*, volume 27, pages 251–260. Wiley Online Library, 2008.

[WK93] Sidney W Wang and Arie E Kaufman. Volume sampled voxelization of geometric primitives. In *Visualization, 1993. Visualization'93, Proceedings., IEEE Conference on*, pages 78–84. IEEE, 1993.

[WMMJ17] Dianhai Wang, Xiaolong Ma, Dongfang Ma, and Sheng Jin. A novel speed–density relationship model based on the energy conservation concept. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1179–1189, 2017.

[WOJK16] Celimuge Wu, Satoshi Ohzahata, Yusheng Ji, and Toshihiko Kato. How to utilize interflow network coding in vanets: A backbone-based approach. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2223–2237, 2016.

[WS11] Kelvin CP Wang and Omar Smadi. Automated imaging technologies for pavement distress surveys. *Transportation Research E-Circular*, (E-C156), 2011.

[WSCY16] Dingwen Wang, Shilei Sun, Xi Chen, and Zhiwen Yu. A 3d shape descriptor based on spherical harmonics through evolutionary optimization. *Neurocomputing*, 194:183–191, 2016.

[YSK$^+$07] Si-Jie Yu, Sreenivas R Sukumar, Andreas F Koschan, David L Page, and Mongi A Abidi. 3d reconstruction of road surfaces using an integrated multi-sensory approach. *Optics and Lasers in Engineering*, 45(7):808–818, 2007.

[ZCEP07] Long Zhang, Wei Chen, David S Ebert, and Qunsheng Peng. Conservative voxelization. *The Visual Computer*, 23(9-11):783–792, 2007.

[ZCK98] Qing-hong Zhu, Yan Chen, and Arie Kaufman. Real-time biomechanically-based muscle volume deformation using fem. In *Computer Graphics Forum*, volume 17, pages 275–284. Wiley Online Library, 1998.

[ZCW$^+$18] Yuming Zhang, Cong Chen, Qiong Wu, Qi Lu, Su Zhang, Guohui Zhang, and Yin Yang. A kinect-based approach for 3d pavement surface reconstruction and cracking recognition. *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[ZD17] Tobias Zirr and Carsten Dachsbacher. Memory-efficient on-the-fly voxelization and rendering of particle data. *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[ZE12] Chunsun Zhang and Ahmed Elaksher. An unmanned aerial vehicle-based imaging system for 3d measurement of unpaved road surface distresses1. *Computer-Aided Civil and Infrastructure Engineering*, 27(2):118–129, 2012.

[ZFS$^+$16] RX Zhong, KY Fu, A Sumalee, D Ngoduy, and WHK Lam. A cross-entropy method and probabilistic sensitivity analysis framework for calibrating microscopic traffic models. *Transportation Research Part C: Emerging Technologies*, 63:147–169, 2016.

[ZGHG11] Kun Zhou, Minmin Gong, Xin Huang, and Baining Guo. Data-parallel octrees for surface reconstruction. *Visualization and Computer Graphics, IEEE Transactions on*, 17(5):669–681, 2011.

[ZGX$^+$17] Yuming Zhang, Steven Garcia, Weiwei Xu, Tianjia Shao, and Yin Yang. Efficient voxelization using projected optimal scanline. *Graphical Models*, 2017.

[Zha02] H Michael Zhang. A non-equilibrium traffic model devoid of gas-like behavior. *Transportation Research Part B: Methodological*, 36(3):275–290, 2002.

[ZHC05] Jian Zhou, Peisen Huang, and Fu-Pen Chiang. Wavelet-based pavement distress classification. *Transportation Research Record: Journal of the Transportation Research Board*, (1940):89–98, 2005.

[ZHH17] Liang Zheng, Zhengbing He, and Tian He. A flexible traffic stream model and its three representations of traffic flow. *Transportation Research Part C: Emerging Technologies*, 75:136–167, 2017.

[ZHWG08] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. In *ACM Transactions on Graphics (TOG)*, volume 27, page 126. ACM, 2008.

[ZMW14] Guohui Zhang, Xiaolei Ma, and Yinhai Wang. Self-adaptive tolling strategy for enhanced high-occupancy toll lane operations. *IEEE Transactions on Intelligent Transportation Systems*, 15(1):306–317, 2014.

[ZS16] Jing Zhao and Shiliang Sun. High-order gaussian process dynamical models for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 17(7):2014–2019, 2016.

[ZSMS14] Yahan Zhou, Shinjiro Sueda, Wojciech Matusik, and Ariel Shamir. Boxelization: folding 3d objects into boxes. *ACM Transactions on Graphics (TOG)*, 33(4):71, 2014.

[ZU17] Feng Zhu and Satish V Ukkusuri. An optimal estimation approach for the calibration of the car-following behavior of connected vehicles in a mixed traffic environment. *IEEE Transactions on Intelligent Transportation Systems*, 18(2):282–291, 2017.

[ZW11] Guohui Zhang and Yinhai Wang. Optimizing minimum and maximum green time settings for traffic actuated control at isolated intersections. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):164–173, 2011.

[ZW13] Guohui Zhang and Yinhai Wang. Optimizing coordinated ramp metering: a preemptive hierarchical control approach. *Computer-Aided Civil and Infrastructure Engineering*, 28(1):22–37, 2013.

[ZWWY08] Guohui Zhang, Yinhai Wang, Heng Wei, and Ping Yi. A feedback-based dynamic tolling algorithm for high-occupancy toll lane operations. *Transportation Research Record: Journal of the Transportation Research Board*, (2065):54–63, 2008.

[ZYW09] Guohui Zhang, Shuming Yan, and Yinhai Wang. Simulation-based investigation on high-occupancy toll lane operations for washington state route 167. *Journal of transportation engineering*, 135(10):677–686, 2009.

[ZZFY18] Yuming Zhang, Guohui Zhang, Rafael Fierro, and Yin Yang. Force-driven traffic simulation for a future connected autonomous vehicle-enabled smart transportation system. *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[ZZSM16] Liteng Zha, Yunlong Zhang, Praprut Songchitruksa, and Danny R Middleton. An integrated dilemma zone protection system using connected vehicle technology. *IEEE Transactions on Intelligent Transportation Systems*, 17(6):1714–1723, 2016.