Transportation Research Part C 154 (2023) 104280

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Using spatiotemporal stacks for precise vehicle tracking from roadside 3D LiDAR data

Yuyi Chang [a], Wen Xiao [b], Benjamin Coifman [c,*]

[a] The Ohio State University, Department of Electrical and Computer Engineering, United States
[b] School of Geography and Information Engineering, and National Engineering Research Center of Geographic Information System, China University of Geosciences, 430074 Wuhan, China
[c] The Ohio State University, Joint appointment with the Department of Civil, Environmental, and Geodetic Engineering, and the Department of Electrical and Computer Engineering, Hitchcock Hall 470, 2070 Neil Ave, Columbus, OH 43210, United States

ARTICLE INFO

Keywords:
LiDAR
Vehicle Tracking
Spatiotemporal Stacks
Highway Traffic
Vehicle Trajectories

ABSTRACT

This paper develops a non-model based vehicle tracking methodology for extracting road user trajectories as they pass through the field of view of a 3D LiDAR sensor mounted on the side of the road. To minimize the errors, our work breaks from conventional practice and postpones target segmentation until after collecting LiDAR returns over many scans. Specifically, our method excludes all non-vehicle returns in each scan and retains the ungrouped vehicle returns. These vehicle returns are stored over time in a spatiotemporal stack (ST stack) and we develop a vehicle motion estimation framework to cluster the returns from the ST stack into distinct vehicles and extract their trajectories. This processing includes removing the impacts of the target's changing orientation relative to the LiDAR sensor while separately taking care to preserve the crisp transition to/from a stop that would normally be washed out by conventional data smoothing or filtering. This proof of concept study develops the methodology using a single LiDAR sensor, thus, limiting the surveillance region to the effective range of the given sensor. It should be clear from the presentation that, provided sufficient georeferencing, the surveillance region can be extended indefinitely by deploying multiple LiDAR sensors with overlapping fields of view.

1. Introduction

This paper develops a non-model based vehicle tracking methodology for extracting vehicle trajectories from 3D LiDAR data. While a traditional model based or feature based tracking system would segment vehicles in each scan and then track the segmented vehicles across scans, this paper reorders the sequence of processing. Specifically, this research seeks to exclude all non-vehicle returns in each scan and retain all features that might come from any vehicle, without attempting to divide the features into individual vehicles. Inspired by the common motion method in Coifman et al. (1998), where the ungrouped vehicle returns are stored over time and the temporal evolution is subsequently used to segment the vehicles from one another. As a result of the temporal component this methodology requires a brief lag of a few seconds between when the data are observed and when the returns can be grouped.

The instantaneous positioning errors are diminished by collecting the time history and incorporating the temporal component into the vehicle segmentation. Perhaps more importantly, many tracking strategies are designed to track in real time, and for these

* Corresponding author.
E-mail address: Coifman.1@OSU.edu (B. Coifman).

https://doi.org/10.1016/j.trc.2023.104280
Received 20 January 2023; Received in revised form 2 May 2023; Accepted 28 July 2023
Available online 4 August 2023

approaches it is necessary to cluster returns in the current scan with those seen in the previous scans. One of the big challenges of such real time processing is the fact that approaching vehicles are first seen far away at the lowest resolution, where it is difficult to group or segment the returns from different road users, as well as from the background. As a result, many real time tracking systems ignore the far field since it is not critical for collision avoidance or other navigational tasks. By compiling the vehicle returns over time, before segmenting the returns into individual objects, the present work can hold off tracking until after a given road user has come their closest to the LiDAR sensor, and thus, has been seen at the highest resolution. At which point, we effectively track backwards in time from the near field to the far field to yield better performance for approaching vehicles.

On a related note, our methodology is robust to the large change in appearance of a vehicle traveling through the surveillance region - typically progressing from a low resolution view of the front, to a high resolution view of the side, and then to a low resolution view of the rear. The set of observable features at the end of its tracking have completely changed from those features seen at the start. Likewise, the centroid of the observed returns shifts from the front of the vehicle to the rear, which if it goes unaccounted for, this shift in viewpoint could be mistaken as actual movement in space that will lead to erroneously low speeds.

1.1. Background

There is an ever increasing need for high quality vehicle trajectory data for the study of vehicle emissions, fuel consumption, and driver behavior. Fuel consumption and vehicle emissions are heavily dependent on acceleration (e.g., Barth et al., 1996; Kuppili et al., 2021; Tsanakas et al., 2022). Whereas most microscopic traffic flow models (typically called car following models) explicitly seek to estimate acceleration (e.g., Chandler et al., 1958; Chen et al., 2012; Jiang et al., 2015; Cattin et al., 2019). While it is trivial to use onboard recording devices to record a host vehicle's acceleration, onboard recording precludes passive observation of traffic. Time series acceleration is very difficult to accurately measure with wayside sensors, remotely measuring time series acceleration requires high resolution wide area detection, which historically meant monocular vision based vehicle tracking.

Most monocular vision based vehicle tracking relies on feature based or blob based detection and tracking. These approaches first find vehicles in each frame, and then associate vehicles between successive frames. The methods were predominantly developed for real time target detection and path tracking for continuity, but the differences in the detection across scans undermines acceleration measurements (see, e.g., Coifman and Li, 2022). Almost all of the publicly available vehicle trajectory data sets were generated from monocular vision based vehicle tracking. Among the vision based studies, the Next Generation Simulation (NGSIM) project is the widest used empirical vehicle trajectory dataset (Kovvali et al., 2007). Recent technological advancements enable collecting data from camera equipped drones. The collections employ low altitude unmanned aerial vehicles (UAVs) combined with computer vision algorithms for vehicle detection and tracking (e.g., Coifman et al., 2006; Barmpounakis and Geroliminis, 2020; Krajewski et al., 2018).

Regardless, monocular vision based vehicle tracking will forever be challenged by the difficulty of precisely positing 3D objects from a 2D image plane into a 2D ground plane (see, e.g., Coifman and Li, 2017 & 2022). There are too many ambiguities in a 2D camera view, e.g., uncertain boundaries between objects and projection errors. Even the conventional projection into the ground plane is problematic since the real world usually is not flat.

LiDAR sensors eliminate the 2D to 3D reasoning necessary for monocular vison based sensing because LiDAR innately measures in 3D space.[1] Still, most LiDAR vehicle tracking frameworks follow the real time detection and tracking strategy: Vehicles are first detected in each scan and then associated across successive scans to complete tracking, e.g., Density-Based Spatial Clustering of Applications with Noise, DBSCAN, (Ester et al., 1996). Deep learning approaches are commonly used in detecting objects of interest in the individual scans (e.g., Guo et al., 2020; Qi et al., 2017). The positions of detected objects are typically represented by bounding boxes. The bounding boxes are connected across scans via an association filter, e.g., JPDA filter (Bar-Shalom et al., 2009). While most LiDAR tracking strategies are real time, there are some approaches that capture returns over time before segmenting individual targets (e.g., El Sallab et al., 2018), as we do in the present study.

LiDAR sensors are a leading perception sensor for autonomous vehicle (AV) technologies, e.g., for threat detection and avoidance (e.g. Chen et al., 2021; Li and Ibanez-Guzman, 2020; Ramasamy et al., 2016) and positioning (e.g., Cheng et al., 2018; Elhousni and Huang, 2020). The AV perception approaches have also been used on instrumented probe vehicles deployed to monitor ambient traffic (e.g., Thornton et al., 2014; Coifman et al., 2016; Wu and Coifman, 2019). Measuring the surrounding environment from mobile LiDAR inherently requires accounting for the ego vehicle motion. The rapidly changing background semantic requires understanding scenes from scan-to-scan.

There has been growing interest in wayside mounted LiDAR sensors for monitoring traffic (e.g., Lee and Coifman, 2015). Most of the wayside applications use a rigidly mounted LiDAR sensor. Compared to mobile LiDAR sensors, the stationary sensor greatly simplifies the task of segmenting vehicles from the background. The most relevant studies use 3D LiDAR sensors mounted next to roadways for vehicle tracking (e.g., Tarko et al., 2016; Zhao et al., 2019; Zhang et al., 2020). While these cited studies employ algorithms for vehicle detection and tracking, their trajectory data are not publicly available. We were only able to identify a few publicly available vehicle trajectory datasets that were collected by wayside LiDAR sensors, with the most prominent being the Ko-PER dataset (Strigel et al., 2014) and succeeding VRU Trajectory Dataset (2022). These two trajectory datasets focus on motion prediction and safety validation, and they were collected at intersections by fusing LiDAR and video data. The Ko-PER dataset contains 340

[1] Stereo vision also innately measures in 3D space, yielding data similar to what 3D LiDAR sensors report. As such, this paper could also be applicable to stereo vision systems. It is worth noting that stereo vision must associate points between the paired views to calculate distance, which degrades performance, whereas LiDAR measures range directly.

vehicles, and VRU dataset contains only pedestrians and cyclists.

1.2. Overview

This paper develops a multi-step method to track road user trajectories as they pass through the field of view of a 3D LiDAR sensor mounted on the side of the road. After a one time calibration to define the right of way, the first step is to exclude all returns beyond the right of way, thus, only retaining the returns from the road users. These returns are stored in a spatiotemporal stack,[2] (or ***ST stack*** for short) on a scan by scan basis without grouping or segmenting the individual returns. Then, we develop a vehicle motion estimation framework to cluster the returns from the ST stack into distinct vehicles and extract their trajectories. Coifman and Li (2022) provides a good background of ST stacks in their original vision based context.

The remainder of the paper is organized as follows. Section 2 presents the data collection and background filtering to yield a clean ST stack containing the moving target returns. Section 3 presents the processing methodologies to extract the vehicle trajectories from the ST stack, including: vehicle segmentation, trajectory extraction, and motion estimation. We close the paper with conclusions in Section 4.

2. Configuration, extraction and stack formation

This research uses a 3D LiDAR sensor mounted near the edge of a road to find and track all of the road users (motor vehicles, bicycles, and pedestrians) in the right-of-way. Collectively, we refer to the area above the road traveled by vehicles as the "right-of-way," or ***RoW*** for short, where the RoW is bounded by the road surface, the curbs, and any blocking in space to exclude fixed objects above the road (e.g., tree branches and traffic signal mast arms). By defining the RoW and road users as we have, all returns in the RoW must come from road users.[3] This section develops the process of extracting and storing the road user returns in the RoW that are recorded in the LiDAR data. Key to segmenting the road user returns from the road is filtering out the background and eliminating the ground returns.

Section 2.1 presents the data collection process, with particular care to address the fact that the appearance of an object changes depending on where it is in the field of view and the limits of resolvability. Section 2.2 explains the process of extracting the road user returns in the RoW from the raw 3D LiDAR data. Section 2.3 discusses how the road user returns in individual scans are assembled into a ST stack to preserve the evolution over time.

2.1. Collecting roadside 3D LiDAR data

Typically, a given 3D LiDAR sensor provides a configurable field-of-view, ***FoV***, and sampling frequency for collecting ranging data. The ranging information for each measurement comes from the time-of-flight. Most commercially available 3D LiDAR are rotational, using mechanical rotating platforms to provide up to 360° horizontal FoV coverage. The vertical FoV coverage varies by sensor model and is defined by the number of laser beam channels, which commonly varies from 16 to 128. The detection range is determined by the sensors, but typically beam divergence limits the effective operating range before reaching the actual limits of the ranging sensor (see, e.g., Tarko et al., 2016). As such, the mounting position is an important factor in determining the resolution of collected data. Mounted next to the roadway, the height and angle of the sensor can be configured to maximize the laser beam coverage in the region-of-interest, i.e., the RoW in this case.

This study develops a method for tracking vehicles from a 3D LiDAR sensor mounted next to the roadway. The methodology is designed to work effectively with a low-end 3D LiDAR sensor, so for this proof of concept study we used a Velodyne "Puck" VLP-16 with 16 vertical scanning angles and a maximum range of 100 m. We found beam divergence, road curvature, and occlusions due to closer vehicles limits the effective range to 50–80 m. This work should be transferrable to any rotational 3D LiDAR with more scanning planes, where additional planes would improve the spatial resolution and potentially expand the effective surveillance region.

The 3D LiDAR data for this study were collected from the roadside along the two-lane Claremont Rd. in Newcastle upon Tyne, United Kingdom on February 22, 2018. Fig. 1.a shows the study site. The VLP-16 3D LiDAR sensor unit scans 360° horizontal FoV and 30° vertical FoV with sampling rate set to 10 Hz. The vertical FoV consists of 16 channels from −15° to 15° at 2° intervals. Fig. 1.b shows how the sensor was mounted on a tripod set about 1.5 m back from the curb, recording at a height of approximately 1 m.

One of the greatest challenges in tracking vehicles in 3D LiDAR data is how much a given target can change in appearance over the FoV. Targets in the near field typically have hundreds to thousands of returns per scan, across multiple vertical scanning angles, coming from two sides of the vehicle, e.g., Fig. 2.b. A target vehicle in the mid field is far enough from the sensor that beam divergence reduces the number of vertical and horizontal scanning angles that the target occupies and the number of returns per scan drops significantly, typically on the order of dozens of returns, coming predominantly from just the front or rear of the vehicle, e.g., Fig. 2.c. A target vehicle in the far field might only appear in a single vertical scan angle and the number of returns might be fewer than 10, at which point the target has few discernable features, and those returns come almost exclusively from either the front or the rear of the

[2] At this point it is sufficient to note that an ST stack is essentially a N+1 dimensional time–space diagram, where there are N dimensions of space and 1 dimension of time. The body of this paper will provide the details of how the ST stacks are implemented in this work.

[3] In extreme cases other objects may appear in the RoW, e.g., animals or debris. The LiDAR data used in this study did not include any such non-road-user objects, but it should be a simple extension of the methods presented herein to track those objects when they are present.

Fig. 1. The study site on Claremont Rd. in Newcastle upon Tyne, UK, (a) aerial photo showing the location of the LiDAR sensor, two signalized pedestrian crosswalks and two directional bus stops (underlying image from Google maps). (b) Photograph of the data collection in progress.

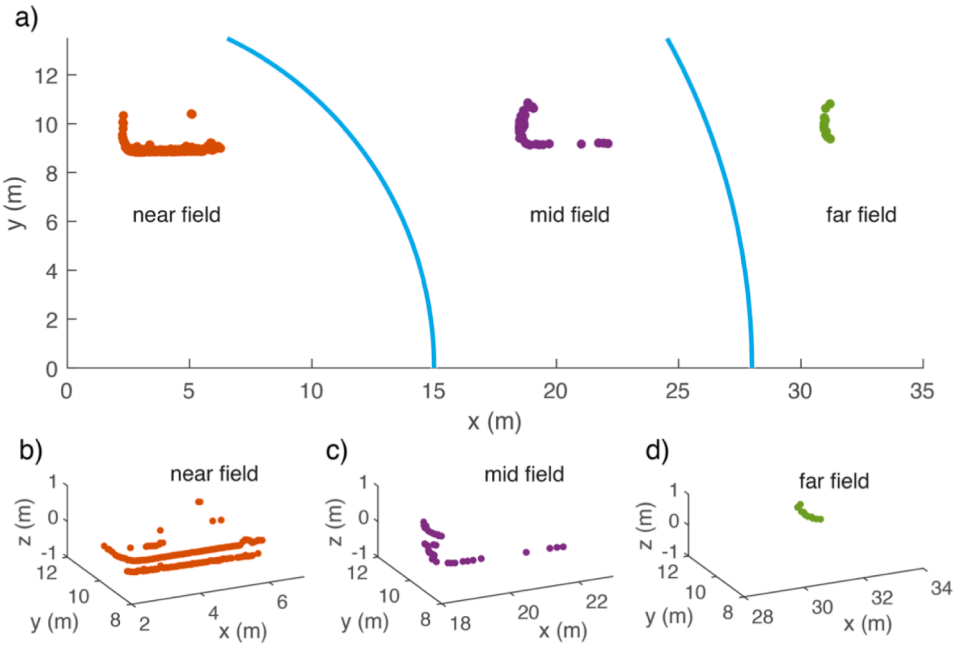

Fig. 2. With the LiDAR sensor at the origin of the plot (a) top down view of the same passenger vehicle as it departs from the LiDAR sensor, illustrating how the appearance changes as it traverses the FoV from the near field to far field. (b)-(d) 3D views of the same data from part a for the three regions: near, mid and far, respectively.

vehicle, e.g., Fig. 2.d. Fig. 2 only shows a departing vehicle, as a vehicle approaches the LiDAR sensor there is a similar progression in reverse order. As a given vehicle passes through the FoV no part of the vehicle is visible over the entire span. As a vehicle transitions from approaching to departing the centroid of the visible region slowly travels from the physical front of the vehicle to the physical rear. If not accounted for, this shift in viewpoint could be mistaken as actual movement in space that will lead to erroneously low

speeds.

2.2. Extracting road user returns

This section presents the methodology to segment road user returns from 3D LiDAR data. The 3D LiDAR sensor captures rich information about surrounding objects, but only the moving targets in the RoW are directly relevant for our work. By definition the roadway is devoid of fixed objects, any large object in the road must be a road user: motor vehicle, bicycle or pedestrian. So, this work (1) defines the RoW boundaries (one time calibration), (2) excludes all returns outside of the RoW (every scan), and (3) identify and exclude road surface returns. All that is left are the returns from the road users. While the main interest in this research is tracking the motor vehicles, the methodology actually tracks all of the road users.

Extracting the returns within the roadway effectively serves as a search space reduction and simplification. This step begins with a pre-defined road boundary. This definition could come from map data or in the long run, we envision developing techniques to automatically identify the road area using an extended period of LiDAR returns. However, since the definition only needs to be made once, for this paper the roadway is manually identified. This manual task is fairly quick to undertake, a human operator simply makes a series of clicks on the point cloud to define the road boundary. The road boundary in the near field is identified by looking at the curb features. Moving further away from the sensor no ground returns are available, so in this region we take the first 30 s of the recorded data, project all available samples to the XY plane, and then the road boundary is derived based on common motion of the aggregated vehicle tracks (see, e.g., Gao and Coifman, 2007). While most of the boundary is defined by the location of the curbs and vehicle travel, the extraction also excludes areas where trees and structures overhang the roadway to ensure that none of these fixed objects are captured. The road boundaries are recorded as a polygon in the XY plane, combined with the blocking z(x,y) to exclude objects overhanging the road, and collectively define the RoW boundaries. At the same time, we identify the lane boundaries within RoW so that the returns and resulting clusters can be assigned to individual lanes.

To illustrate the processing, consider the data from a single scan in Fig. 3.a along with the pre-initialized roadway boundaries indicated with solid lines. Note that throughout Fig. 3 each part is split into two views, (i) a 3D plot and (ii) the exact same data projected to a 2D plane. Using the boundaries from Fig. 3.a, all of the returns are classified as either being in-road or beyond the road boundaries, yielding Fig. 3.b where the returns are now shaded based on their classification. The returns within the road boundaries are further classified as being above the road or on the road surface, as shown in Fig. 3.c. The road surface returns are discarded, while all returns above the road must come from road users and are retained as such, Fig. 3.d.

To distinguish between above or on the road surface we use a road model constructed from an ensemble of possible road pavement points identified from multiple trials. The first step is to sample the input data by capturing all of the in-road returns over a fixed duration (20–50 scans in our case). Then, a plane is fitted to the selected sample using MLESAC algorithm (Torr et al., 2000). Since the road pavement is relatively flat, we configure this feature selection algorithm to look for a plane that has normal vector $n = [0, 0, 1]^T$. A point will be declared as an inlier point to the fitted plane if it is within $\pm 0.2m$ of the plane. We refer the inlier points as pavement return candidates. These processes are repeated five times using independent time windows, yielding an ensemble of pavement return candidates. Next, we perform a group voting within the ensemble to identify the common scan angles that appear in all of the candidates. This group voting step is designed to mitigate the bias in the pavement identification process where a return coming from a road user is unlikely to appear in other sets of candidates. Finally, we generate a road model by fitting the remaining samples to a 2D interpolant $z = F(x,y)$. The road users are separated from the road pavement by comparing the sample's vertical coordinate, z, to the local elevation of the road, $\hat{z}$, estimated from the interpolant F given the sample's x and y. In practice, we add a safety margin of $d = 0.1m$ to $\hat{z}$. In this case, given a point $p = (x,y,z)$, p is classified as a road user return if $z > \hat{z} + d$ where $\hat{z} = F(x,y)$.

2.3. ST stack formation

After discarding all of the returns outside of the RoW in a given scan, as per Section 2.2, all of the remaining returns are assumed to come from road users. This section presents the mechanism for storing the ungrouped road user returns over time. Returning to the example in Fig. 3, we left off finding the road user returns from a single scan in Fig. 3.d. In particular, Fig. 3.d.ii, with all of the road user returns projected into the XY plane. We repeat this process in every successive scan and save the results from each scan as a plane indexed by time in a new 3D coordinate system, (x,y,t). Illustrating this process for the time window spanning 17 successive scans yields Fig. 3.e.i for the (8 scans before to 8 scans after the single scan in Fig. 3.d.ii using a different color for each scan). In other words, for each scan we project all of the road user returns into the ground plane and store this planar result at the scan time and at this stage no effort is made to segment the vehicle returns into distinct objects.

More formally, Fig. 3.e.i is an example of a ST stack showing how the set of road user returns in the XY plane evolve through the FoV over time. For context, ultimately, we want to group all of the returns from a given road user over time, and segment different road users from one another, as shown with a unique color for each road user in Fig. 3.f. But this grouping and segmentation is only done after all of a given target's information has been extracted, as will presented in Section 3.

3. Target segmentation- from the ST stack to trajectories

This section presents the process of automatically grouping the returns into distinct road users and when necessary, segmenting the clusters to isolate the 3D band of returns from each individual road user. After this grouping and splitting the individual road users are

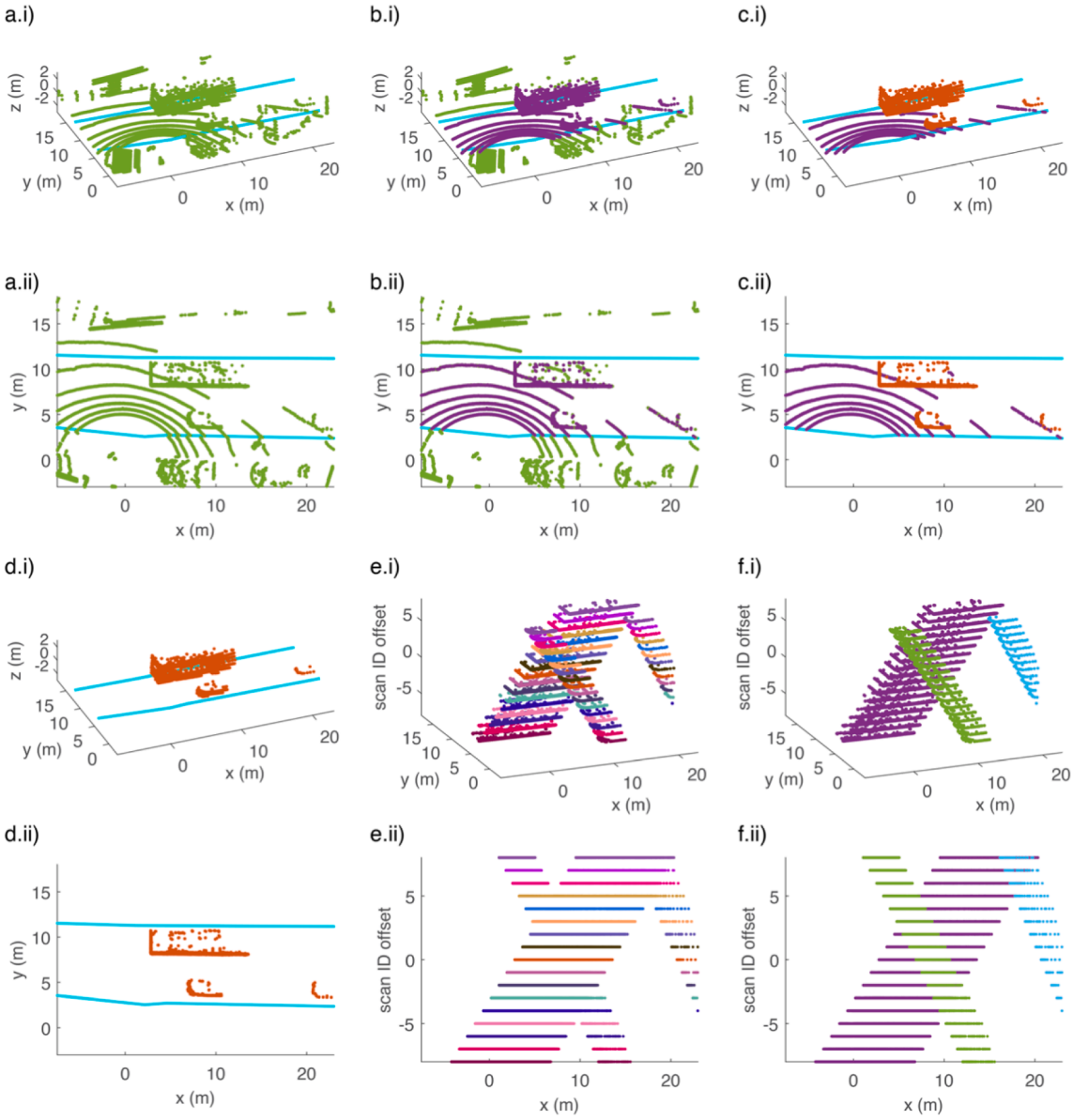

Fig. 3. Illustration of the processing steps for obtaining the ST stack from individual 3D LiDAR scans. Parts a-d each show two subplots: (i) in (x,y,z) space and (ii) the same data in (x,y) space. (a) A single scan of 3D LiDAR from the input data; (b) background filtering using the predefined road boundary; (c) identifying road users using the iterative ground filtering algorithm; (d) the resulting road user returns after the proceeding steps. Parts e-f also show two subplots, but now (i) in (x,y,time) and (ii) the same data in (x,time). (e) Constructing a ST stack by stacking the scans in the XY plane along the time axis; (f) the same data after our series of clustering operations to identify the individual vehicle tracks. Note that the data in part d.ii are shown at scan ID offset zero in parts e and f.

tracked to extract trajectories that are robust to the changing appearance, e.g., transitioning from the front to the rear of a target vehicle as it passes the LiDAR sensor. For this study the processing is not undertaken until a given target leaves the FoV, so this work can only be done in "near real time" after a small time lag.

This work is concerned with movement along and across the road rather than in pure Cartesian space. So, the processing is simplified by projecting the 2D XY coordinates to a new 2D space that is based on the road geometry. Specifically, to effectively associate returns to the correct lane we define a coordinate system such that distances can be measured along the roadway, s, and across it, w. This roadway coordinate system (RCS) provides a unified position representation system such that both longitudinal and lateral positions can be related across lanes. The RCS is constructed as a sequential series of linear segments that follow the centerline of the road as the road curves. The junction between two successive segments is trimmed so that the corners are replaced with arcs to

smooth out the gradient to ensure the continuity of the coordinate system. The projection from the original 2D XY coordinates to RCS is performed by a series of vector projections. For a given scan at time *t*, the samples are first associated to the corresponding segments based on proximity. The data points are projected via point-to-line orthographic projection, and longitudinal distance, *s*, is measured from the segment starting point to the projected point plus any segment lengths that have passed. The lateral position, *w*, is then calculated as the Euclidean distance from the given point to the projected point (i.e., measured along the normal vector). We will use distance in RCS throughout the remainder of the paper to express position relative to the road geometry.

The details of grouping and segmenting the returns in the ST stack are presented in the following subsections. Section 3.1 searches the ST stack for continuity to group returns coming from the same target, separate from all other targets, and doing so across short occlusions. Section 3.2 develops trajectory extraction methods.

3.1. Segmenting targets in the ST stack

This section develops the method of segmenting road users from the ST stack. The result is a new ST stack that contains the spatiotemporal information of all moving objects in the RoW. This section presents the processing to go from the raw ST stack in Fig. 3.e to the distinct road users in Fig. 3.f. Each moving object creates a ***track*** in the ST stack, typically taking the form of a tube in the 3D stack. Moving to a larger example, Fig. 4.a shows 23 tracks from road users: 12 far lane motor vehicles moving from top to bottom, 7 near lane motor vehicles moving from bottom to top, and 4 pedestrians crossing the road and thus, barely travel any distance s along the road. This section will segment the ungrouped returns in Fig. 4.a to the distinct road users sorted by direction of travel, e.g., the far lane tracks in Fig. 4.b. A given object may be split across more than one track if it is momentarily occluded or it is far away from the LiDAR. On the other hand, if two objects get close together it may become difficult to separate the two individual tracks. This processing is split into three steps: Section 3.1.1 identifies short-term connectivity with returns belonging to the same object; Section 3.1.2 seeks long-term connectivity to join tracks arising from a single object, and bridge occlusions and brief interruptions when a vehicle is far from the LiDAR sensor; finally, Section 3.1.3 classifies all tracks to distinguish vehicles that travel along the road (motor vehicles and bicycles) from pedestrians and some bicycles that travel across the road.

3.1.1. Volumetric neighborhood clustering

This work uses a Volumetric Neighborhood Clustering, ***VNC***, (Chen et Al., 1998) to segment distinct tracks in the ST stack. The VNC consists of two parts: first voxelization by turning the ST stack into voxels, and then identifying connected components across voxels to form tracks. Fig. 5 illustrates the fundamental process in a simpler 2D toy example. Fig. 5.a shows the input data. The space is partitioned into a uniform grid of boxes, and each box stores a 1 if there are any returns in its area and 0 otherwise (Fig. 5.b). The next step is to perform neighborhood clustering where two voxels have a shared edge[4] and both marked with 1 (Fig. 5.c). Finally, the connections made in the binary matrix are transferred to the returns corresponding to each cell to complete the volumetric neighborhood clustering (Fig. 5.d).

Switching to the actual implementation, we apply the method to the 3D ST stack in the RCS, so what were 2D boxes in the hypothetical example of Fig. 5 become 3D voxels. For this work the voxelization resolution is set so that the ST stack is partitioned into a uniform 3D binary grid of 0.1sec $\times$ 0.1 m $\times$ 0.1 m voxels. The longitudinal range spans -80 m to $+$ 80 m, and lateral range spans -5 m to $+$ 5 m.

The base VNC works well when the returns are close together (near field and mid filed), but in the far field the gaps between returns in a given track can grow too large to be merged using the base grid resolution. Likewise, occlusions anywhere in the FoV can split a given track beyond the range of the base resolution. To this end, we boost the connectivity using a 3D cuboid dilation kernel, as discussed below. After the dilation, neighborhood clustering is performed by searching the connectivity among voxels. A given voxel containing returns is connected to another voxel containing returns if they share a corner, edge or face. A unique ID number is assigned to each independent cluster, and these ID's are projected back to the returns in the original ST stack. We process the stack in chunks of 100 sec (1000 individual scans), but in practice the method could be implemented on a rolling window that processes the clusters as soon as they are closed.

Road users are sorted by their direction of travel, with road users that travel along the road being taken as vehicles and road users that travel across the road as pedestrians.[5] To this end, we employ a two-stage VNC procedure to identify vehicles within the ST stack. This consists of a preliminary VNC, ***pVNC***, followed by a main VNC, ***mVNC***. The two VNC operations follow the same process except that each uses a different dilation strategy. The pVNC performs a "discover scan" on the input ST stack. The clusters within the ST stack are identified using a small, generic dilation kernel sized 4x10x5 (for *t*, *s* and *w* axes, respectively). At which point, we remove the clusters that appear to be crossing the road, as follows. A cluster is excluded from the pVNC if the lateral distance is greater than 4 m (50% of the lateral span of the road). Those clusters arise from pedestrians and bicycles crossing the road and transient returns from the interior of vehicles via window *shoot-through*. By removing the non-vehicle clusters, a valid vehicle cluster is less likely to be joined with a noise cluster, thus improving the robustness of the algorithm. In practice, these cross-road movements can be retained and processed separately to track the pedestrians.

Next, for the along the road movements we perform the mVNC to the remaining ST stack after removing the non-vehicle clusters.

[4] Note this 2D example does not consider shared corners as connecting, but the full 3D version includes shared corners for connectivity.

[5] Note that pedestrians walking on the sidewalk are outside of the RoW by definition, and depending on their motion, a bicycle could wind up in either category.

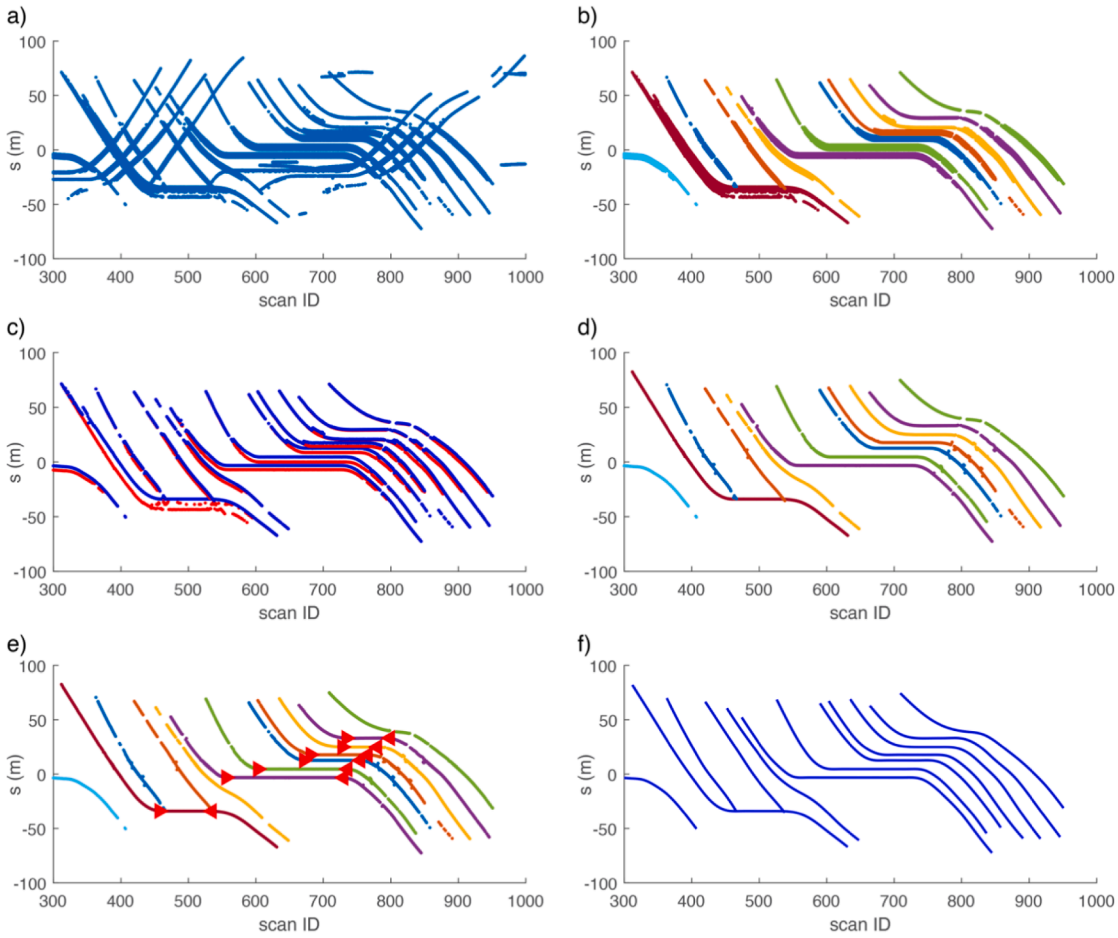

Fig. 4. An example illustrating the individual steps for obtaining vehicle trajectories from raw LiDAR data showing (a) the ST stack projected into time and distance along the road, s, (b) track clustering and association (for clarity only showing far lane), (c) leading edge and trailing edge trajectory extraction, (d) shifting the trailing edge by the vehicle length to align and fuse with the leading edge to generate rough vehicle trajectories. Note an overtaking maneuver around scan 500, and a stop wave due to red light starting at scan 550 onward. In order to preserve crisp transitions to and from stops, (e) stop detection with stop beginning and ending times denoted by right and left pointing triangles, respectively, and (f) the final vehicle trajectories. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

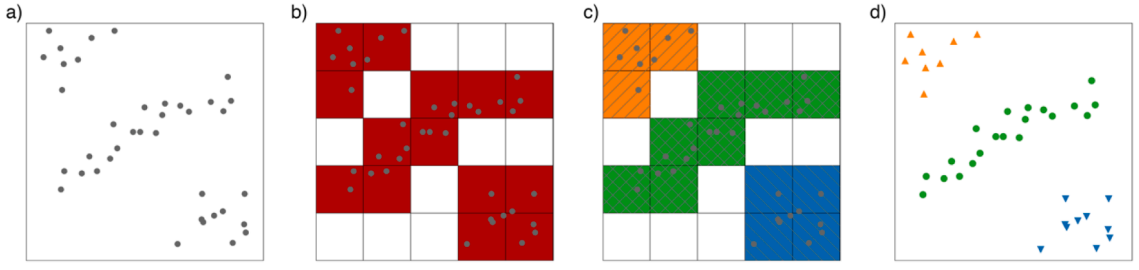

Fig. 5. A hypothetical 2D example of volumetric clustering showing (a) the raw input data in continuous domain; (b) discretized data and volexization matrix; (c) clustering by nearest neighbor; (d) the resulting clusters.

The mVNC uses an adaptive dilation based on the longitudinal distance from the LiDAR sensor, where the adaptive dilation kernel size increases with distance. The dilation size is 4x10x4 (t, s and w axes, respectively) for $|s|\langle 12m$, 5x13x4 for $|s|\langle 20m$, 7x14x5 for $|s|\langle 30m$, and 9x16x5 for anything beyond 30 m.

3.1.2. Resolving overgrouping

Occasionally two vehicles are merged during the mVNC due to the size of the dilation. To catch these errors after the VNC clustering, we evaluate a given cluster's measured length along the road, Δs, in each scan throughout the entire time history of the given cluster. For the n-th scan the cluster's measured length, $\Delta s(n)$, is given by Equation (1), which is simply the instantaneous longitudinal span of the cluster in the RCS. We also evaluate its rate of change, $r(n)$, via Equation (2). Then we calculate the maximum and the 97th percentile measured lengths of the cluster across all scans, Δs_{100} and Δs_{97}, respectively. This cluster will be considered an overgrouping event if either Equation (3) or Equation (4) is met, where γ_1 is 6 m and γ_2 is 0.05

$$\Delta s(n) = s_{max}(n) - s_{min}(n) \tag{1}$$

$$r(n) = \Delta s(n) - \Delta s(n-1) \tag{2}$$

$$|r(n)\,|\rangle \gamma_1 \tag{3}$$

$$R = \frac{\Delta s_{100} - \Delta s_{97}}{\Delta s_{100} + \Delta s_{97}} > \gamma_2 \tag{4}$$

To address a given overgrouping event we reformulate the problem to find the shortest path between the furthest nodes in a directed graph. We break the given cluster into a large number of mini-clusters by rerunning VNC strictly on the returns in the overgrouped cluster using a constant but small dilation kernel of 3x3x2. This small dilation kernel ensures that almost all previously overgrouped objects will be separated. Then, we construct a directed graph where the nodes are individual mini-clusters, and the links are defined using the following cost function. For node i and j, the cost moving from i to j is defined by Equation (5), where N is a gating window that is empirically set to 10 successive scans (1 sec). The quadratic term Δn^2 enforces time continuity, discouraging the algorithm to skip a prolonged time period between the clusters when moving from one end to another. The graph is directional to ensure that a vehicle can only travel forward in time, and thus, the successive mini-clusters must also progress forward in time, so given clusters i and j, a finite $c(i,j)$ will only result if cluster i appears earlier than cluster j. We define the origin (source) and destination (drain) by finding the terminal nodes within the graph (i.e., a node is defined as terminal node if it strictly has paths in, or strictly paths out). The shortest path from the origin to destination implies a minimum cost traversing all nodes in between. After removing the nodes within the shortest path from the graph, the remaining nodes are either from another vehicle, an overgrouping of more than one vehicle, or noise. The remaining min-clusters are rerun through the processing in Section 3.1.1, checked for possibly crossing the road, and then rechecked for overgrouping.

$$c(i,j) = \begin{cases} 20\Delta n^2 + \Delta s + 20\Delta w, & 0 < \Delta n < N \\ \infty, & otherwise \end{cases} \tag{5}$$

where,

$$\Delta n = min\left(n_j\right) - max(n_i)$$

$$\Delta s = s_j\left(min\left(n_j\right)\right) - s_i(max(n_i))$$

$$\Delta w = w_j\left(min\left(n_j\right)\right) - w_i(max(n_i))$$

3.1.3. Cluster association

While at this point in the processing most vehicles will correspond to a single track, occlusion and beam divergence might oversegment a track, splitting it into multiple individual clusters. This section seeks to identify the presence of oversegmentation and join the separate clusters that belong to the same vehicle track. For vehicles traveling in the same lane within a conservated road it can be safely assumed that those vehicles originated from a common starting point (source) to a common ending point (drain). If a given vehicle is seen throughout the sensor FoV, then there exists a unique path from the source to the drain, representing the spatial–temporal presence of the vehicle.

To start with, we take all clusters within a given lane. The lateral position of each return is already recorded in the RCS and is now used to determine which lane it is in. A vehicle may change lanes due to passing movements or lane change maneuvers. We use the range of lateral position in each scan to determine which lane the given cluster is in at that instant. Using the lane boundaries from Section 2.2, each scan a given cluster is assigned to the individual lane in which the majority of returns fall into. In our case there are only two lanes: the near lane with traffic traveling from right to left along the road, corresponding to increasing longitudinal distance in the RCS, and the far lane with traffic traveling from left to right along the road and longitudinal distance decreasing in the RCS.

Within the given lane, a directed graph is constructed consisting of N nodes where N is the number of clusters from the previous step. Then, for any two nodes i and j, we check if there is a path that exists. To this end, we iterate all nodes, and for each node N_i, the ending scan ID and s are identified. A gating window is set to –5 to 25 scans from the ending scan ID, which corresponds to 0.5 sec

backward in time when the current node ends and 2.5 sec forward in time. The gating window for s is determined by the direction of travel, as encoded by the lane, the s ranges from –5m to 25 m for the far lane where vehicle travel with increasing s, and –25 m to 5 m for near lane where vehicles travel with decreasing s. The starting t and s from all nodes, except the current one, are compared to the window. Those within the windows are referred to as possible matches, thus, if there are M possible matches, then there will be M outgoing paths from the N_i.

Next, for a given cluster N_i and each possible match M_j, to quantify the smoothness of the trajectory formed by two clusters we calculate a pairwise matching score as follows. A possible trajectory, PT, is found using spline interpolation to concatenate the two clusters over the time gap between them. Then, the pairwise matching score $w(N_i, M_j)$ is defined as $var(PT^{'})/mean(PT^{'})$ where $PT^{'}$ is the first derivative of the fitted PT within the transition region. Since the clusters N_i and M_j might come from different parts of the same vehicle, care must be taken when the viewpoint changes in the near field where vehicles transition from approaching (front view) to departing (rear view). If the two clusters are strictly upstream of the LiDAR sensor at $s = 0$, the leading edge of both clusters will be used if they are both strictly downstream the trailing edge will be used. If the transition region between the last position of N_i and first position of M_j falls in or spans -5 m to $+5$ m, then the pairwise matching score is found for both the leading edge and separately the trailing edge and an average the two matching scores will be used. In any event, path weights are checked after all nodes are traversed. Two associated nodes are merged for those paths whose weights are below a threshold $\epsilon = 0.035$. The process is repeated in the given lane until no more nodes can be matched.

Fig. 6 shows a sample of the final tracks that result after clustering all of the objects. It is difficult to visualize the tubular 3D tracks in a 2D figure, so this figure breaks it down into a few parts. First, the left side of Fig. 6.a shows the clustered vehicle returns from a

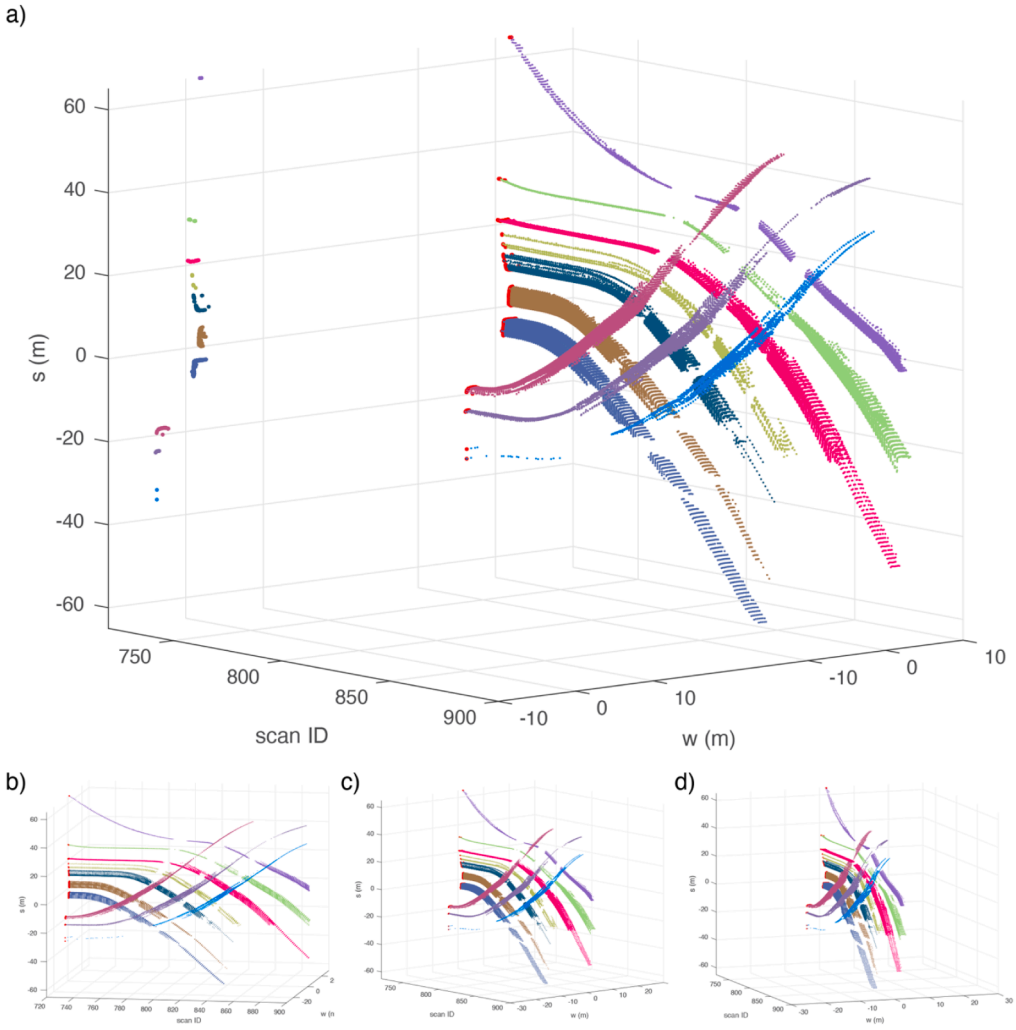

Fig. 6. (a) The left side of this plot shows the clustered vehicle returns in scan 720 for three near lane vehicles (negative w) and seven far lane vehicles (positive w), the right side of the plot highlights the same returns but then adds the associated tracks over the next 180 scans (18 sec). (b-d) repeat the same plot from different viewing angles to highlight the 3D nature of the tracks.

single scan (scan 720), similar to Fig. 3.d.ii only without showing the road boundaries and now the direction of travel is along the vertical axis of the plot. The clusters with a negative lateral position, w, are in the near lane and the positive w are in the far lane. Looking at this single scan it is difficult to see the 10 distinct vehicles because at this instant eight of them are stopped at the signalized pedestrian crosswalk. At jam density like this, the distance between clusters is shorter than the individual cluster link. The right side of Fig. 6.a starts with the returns from scan 720 on the left side of each track, but then adds the associated tracks over the next 180 scans (18 sec). This plot is similar to Fig. 3.f only rotated so that now the time axis is horizontal. Note that the same range on the w axis is repeated for the left and right halves of Fig. 6.a. Now one can clearly see the tracks from the three near lane vehicles and seven far lane vehicles pull apart as they accelerate away from the signal in their respective directions. Note that the most upstream near lane vehicle is largely occluded by the two vehicles in the same lane that come between it and the LiDAR sensor until scan 785. Similarly, all of the far lane vehicles show brief gaps when the near lane vehicles pass between them and the LiDAR sensor. These line of sight occlusions are unavoidable with a single sensor but can be addressed with additional sensors to provide overlapping coverage from different angles. Regardless, to highlight the 3D nature of the tracks, Fig. 6.b-d repeat the same plot from different view angles (this time omitting the presentation of the single scan).

3.2. Extracting vehicle trajectories

This section takes the road user tracks classified as vehicles from Section 3.1 and extracts the trajectory information from the vehicle tracks with respect to the RCS. Since most tracks transition from a front view to a rear view, this extraction includes extracting the leading and trailing edges of a given vehicle and then fusing them by its length. These steps are presented in Section 3.2.1. Section 3.2.2 seeks to improve the accuracy of a given trajectory during stop-and-go movements, where conventional filtering techniques would blur the transition to/from a stop and then Section 3.2.3 estimates the vehicle motion using a RTS smoother to produce the final vehicle trajectories. Section 3.2.4 presents the results of the vehicle tracking.

3.2.1. Position referencing

As noted previously, the appearance of a given vehicle changes dramatically as it traverses through the sensor FoV. The changing viewpoint results in motion around the vehicle captured in the LiDAR returns that is unrelated to the actual motion of the vehicle through space, and so the motion from the changing viewpoint needs to be removed from the final trajectory of each vehicle. This work will use the front of each vehicle as a consistent reference point for all trajectories in the given lane. Fortunately, for the roadside LiDAR sensor deployment the transition is smooth from vehicle front to vehicle side (with both front and rear visible) to vehicle rear. So, for each vehicle we define a raw leading edge trajectory, consisting of the set of the downstream-most points in the cluster at each time step, and a raw trailing edge trajectory, consisting of the upstream-most points in the track at each time step. When the vehicle is roughly centered at the sensor, both leading and trailing edges represent the true front and rear of the vehicle and they are visible through the vehicle side body, providing a handoff area for fusing the two edges.

To estimate the offset between the leading and trailing edges we measure the distance between the leading and trailing edge trajectories while the vehicle is in the near field. For this offset estimation the near field is defined as being the range where both the leading and trailing edges are longitudinally within $\pm$ 12 m along the RCS relative to the LiDAR sensor. The final offset is then estimated using a minimum mean absolute error estimator from the set of near field measurements of the given vehicle and this distance is recorded as the vehicle's length. This length measurement excludes any scans where the vehicle might be occluded, as evidenced by the end of the track transitioning to the end of a closer track at the next scan angle for the given instant.[6]

Given a vehicle track, e.g., Fig. 7.a, for each scan we identify the leading and trailing vehicle position in the longitudinal coordinate distance, s. Consider the example vehicle as it passes through the range of accepting both the leading and trailing edges, Fig. 7.b projects all of the returns from this vehicle in scan 399 into the 3D lateral, longitudinal and z space of the RCS. This instantaneous plot shows the leading and trailing edges correspond respectively to the front and rear bumpers of the vehicle.

The Fig. 7.c shows the front and rear tracks (leading and trailing edges) from part a as well as the regions where the leading and trailing edges are accepted. At this point the leading and trailing edges are clipped, as shown in Fig. 7.d, with the leading edge retained from the start of observation until it reaches the far end of the near field at $s = -12m$ and the trailing edge is retained from the start of the near field at $s = 12m$ until the vehicle departs the field of view. The retained trailing edge is shifted downstream by the vehicle's final offset. The result for the sample vehicle is shown in Fig. 7.e. At this point the two edges are combined into a single "raw trajectory." For the short period of time while the vehicle is in the near field, at each time step there will usually be two positions recorded in the raw trajectory (one from the leading edge and one from the shifted trailing edge). Except when there is a transient error (e.g., an undetected occlusion) these two positions will be very close together. Regardless, for vehicles traveling in the opposite direction the near field boundaries are swapped, with the rest of the processing is unchanged. This processing can be seen on a larger scale in Fig. 4.c-d.

[6] In rare cases the occlusion might last long enough that it is not possible to measure the true vehicle length, in which case the vehicle's position, speed, and acceleration can still be measured, but its exact length is not measured. These extreme occlusion events are flagged and processed separately. The main impact when this situation arises is that when these vehicles are viewed from the rear the location of the front will not be measured.

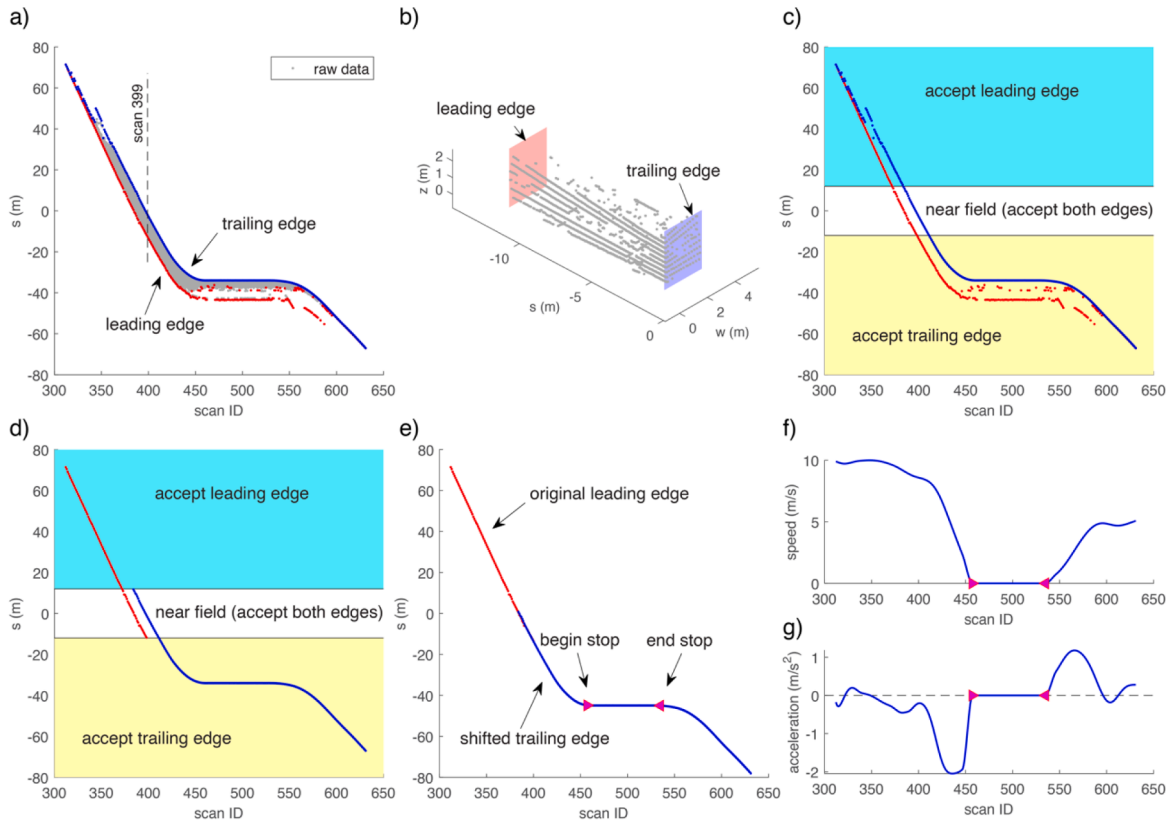

Fig. 7. Trajectory extraction from vehicle track: (a) identify leading and trailing edges from the input vehicle track; (b) for illustration, the leading and trailing edges projected back to the original 3D at scan 399; (c) before and (d) after applying predefined acceptance zones to truncate the leading and trailing edges; (e) shift the trailing edge by the measured vehicle length to prepare for fusing and stop detection. The resulting (f) speed and (g) acceleration after trajectory fusing and motion estimation.

3.2.2. Stop detection

Almost all smoothing processes are akin to a low pass filter: removing short duration fluctuations and retaining persistent trends. However, there are a few abrupt changes that we want to preserve, specifically the transition to and from a stop. A simple smoother will average across these transitions, distorting the actual motion of the vehicle. So before smoothing the data, this section finds the stop transitions in the raw data so that we can adjust the smoothing to preserve the crisp stop transitions.

When a vehicle stops, the time series longitudinal position should be constant, with the trajectory exhibiting a flat line in the time–space plane, perhaps with minor fluctuations due to noise or small vehicle *move-up* while in a queue. Examining a given raw trajectory in the time–space plane this section explicitly seeks horizontal line segments with the goal of establishing their start and end points, e.g., respectively the curve and the triangular end points in Fig. 7.e. To this end, the following stop detection processing is done separately to each raw trajectory. We adapt the conventional line detection problem by modifying the detector into a horizontal line finder using RANSAC algorithm (Fischler and Bolles, 1981). The detector's fitting function uses standard linear regression. We define the selected feature C to be a subset of the raw trajectory position that has roughly a constant value as a function of time. The algorithm essentially is looking for a large number of returns within a small threshold distance of a target horizontal reference line. Throughout the entire track, the RANSAC algorithm classifies each time step as to whether or not it meets the criteria. The goodness of fit for C is quantified by the evaluation function in Equation (6), where s_n is vehicle position, β_0 is intercept term, and β_1 is the slope. In addition to the fitting errors expressed in mean square form (MSE), we regularize the slope of the fitted line β_1 by adding a penalization term w_1. Generally, a single stop is characterized by many successive samples meeting the criteria, potentially with a few intermediate samples that fail to meet the criteria. So, for the sake of our work, we take the first and last instances that meet the criteria to bound the stop. If a vehicle has come to a stop, the detector will provide the scan ID when the stop begins/ends and the location where it occurs. Note that the RANSAC algorithm will only identify a single stop event per pass, so after identifying one stop event, we remove those data from the track and rerun the RANSAC to see if there is another stop event, and iterate until no more stops are found. This processing can be seen on a larger scale in Fig. 4.e.

$$C = w_1\beta_1^2 + \sum_{n=1}^{N}(s_n - \hat{s}_n)^2 \tag{6}$$

$$\hat{s}_n = \beta_0 + \beta_1 t_n$$

3.2.3. *Trajectory fusing and motion estimation*

At this point the raw trajectory has two sources of noise: potentially conflicting positions while both edges are retained in the near field and any original measurement jitter from the leading or trailing edges. A Rauch–Tung–Striebel, ***RTS***, smoother (Rauch et al., 1965) is used to simultaneously perform trajectory fusing between the leading and shifted trailing edge trajectories while also undertaking the motion estimation to extract the speed and acceleration of the fused trajectory. The RTS smoother is essentially a noncausal version of the Kalman filter consisting of a forward and a backward pass. First, the forward pass acts similarly as a standard Kalman filter except the state covariances are preserved for subsequent use in the backward pass. The trajectory fusing is performed during the forward pass where we use position recorded from the original leading edge and shifted trailing edge when the vehicle is within the near field hand-off area defined in Section 3.2.1. The smoothing is done during the backward pass, it corrects the fused trajectory and the initial guess of vehicle speed and acceleration based on the posterior statistics obtained from the forward pass. Hence, the RTS smoother achieves a better tracking accuracy compared to a standalone Kalman filter. We configure the state transition matrix F to implement a 1D constant acceleration motion model (Equation (7). To smooth the data, following Bar-Shalom et al. (2004) a process noise matrix Q is employed using a discrete white noise model (Equation (8) where T is the sampling interval, and σ^2 is the noise variance. This work sets σ^2 to 0.2. The output trajectories contain 1D information including smoothed position, speed, and acceleration.

$$F = \begin{bmatrix} 1 & T & \frac{1}{2}T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

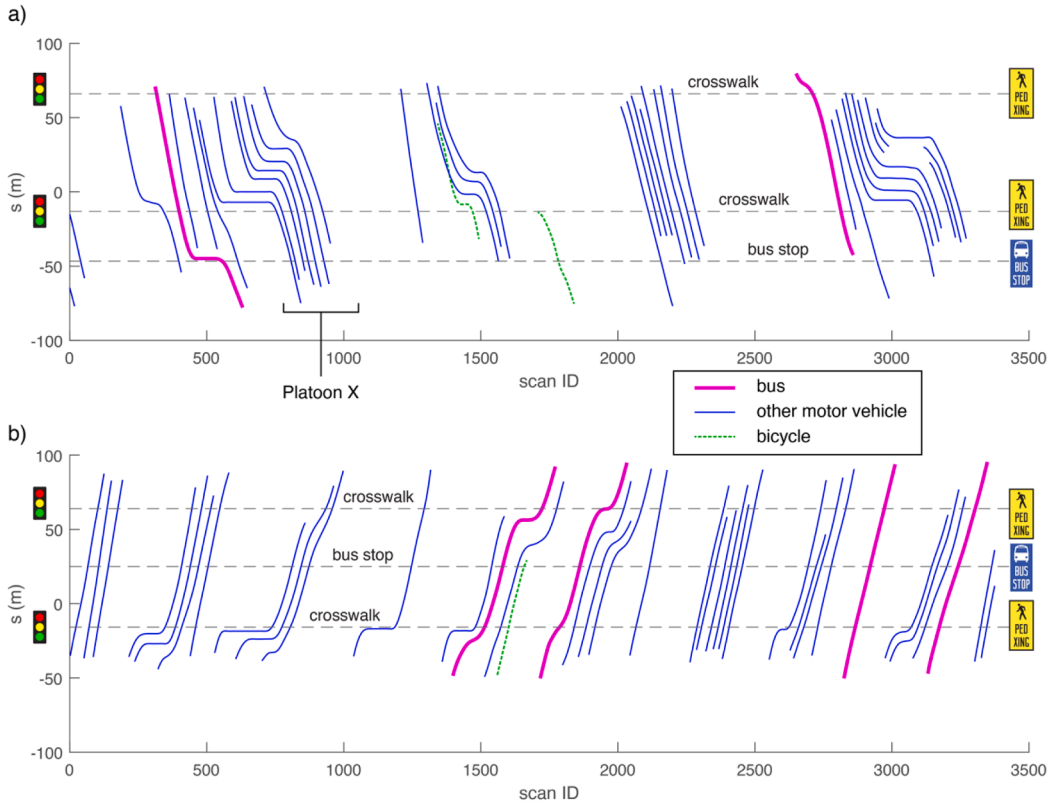

Fig. 8. Time space diagram of the final trajectories, (a) far lane with 38 vehicles and (b) near lane with 37 vehicles. Note that the bus stops are at different locations for the two directions.

$$Q = \begin{bmatrix} \frac{1}{4}T^4 & \frac{1}{2}T^3 & \frac{1}{2}T^2 \\ \frac{1}{2}T^3 & T^2 & T \\ \frac{1}{2}T^2 & T & 1 \end{bmatrix} \sigma^2 \tag{8}$$

At this point, if the raw trajectory came to a stop, then the fused trajectory will have smoothed over the stop and start events. Our processing uses the stop handling procedure in Coifman and Li (2017). Specifically, we define an "almost stopped" threshold that is applied to the fused trajectory, $v_T < 1.2m/s$. Whenever the raw trajectory comes to a stop (as per Section 3.2.2), for this study, we identify the stopped position based on the median position within the stop period and set all positions within the stop period to this fixed position. For the ongoing example in Fig. 7.e the beginning and ending times of the stop period are indicated with triangles. Next, the algorithm (1) sets speed and acceleration to zero beginning at the stop time, (2) finds the last instant, n_c that the fused trajectory was above v_T, and (3) fits a spline curve between the fused trajectories prior to n_c and the forced stop from #1. Likewise, whenever the raw trajectory resumes moving, the algorithm (4) sets speed and acceleration to zero up to the start time, (5) finds the first instant, n_c, that the fused trajectory goes above v_T, and (6) fits a spline curve between the start time from #4 and the fused trajectory after n_c. Fig. 7.f-g show the resulting speed and acceleration for the ongoing example, while Fig. 4.f shows the fused trajectories in the larger example.

It is also important to note that whenever a vehicle is dropped and then reacquired after a few scans due to occlusion or other interruptions, the position and speed over the intervening scans can be interpolated from the motion estimation results. In this study we apply this interpolation for gaps up to 20 successive scans (i.e., 2 sec) and then whenever an interpolation was made, manually inspect the results for quality assurance. If an error is evident, we mark the samples as "invalid trajectory." Of course, a more conservative approach would be to simply mark all unseen samples as "invalid trajectory," without any interpolation, as would be appropriate for some applications.

3.2.4. *Final trajectories*

This proof of concept study uses a 6-minute dataset containing 75 vehicles (including 3 bicycles) and the methodology tracked all 75 vehicles that passed in the LiDAR dataset. Meanwhile, as noted previously, the pedestrians crossing the road are excluded from our processing but could be retained for tracking. Fig. 8 shows a time–space diagram of the final trajectories. Within the LiDAR FoV the site includes two signalized crosswalks and one bus stop in each direction. Vehicle queues are readily apparent upstream of the signalized crosswalks in both directions, with about 30% of the vehicles coming to a stop while in the FoV. Fig. 8.a shows the tracking is not perfect, as two far lane vehicles in a dense queue are completely occluded for a brief period around scan 3000, and thus, not tracked across the gap. Depending on the application one might want to retain these gaps and strictly rely on the observed data or employ methods that impute missing trajectory data (e.g., Sazara et al., 2017). Fig. 8.a also shows two trajectories that seemingly terminate prematurely near scan 500. Because this plot does not show the lateral motion it is not clear in this figure that the two vehicles partially cross the center line to pass a stopped bus that is blocking the road at scan 500 while it services passengers. These two vehicles that

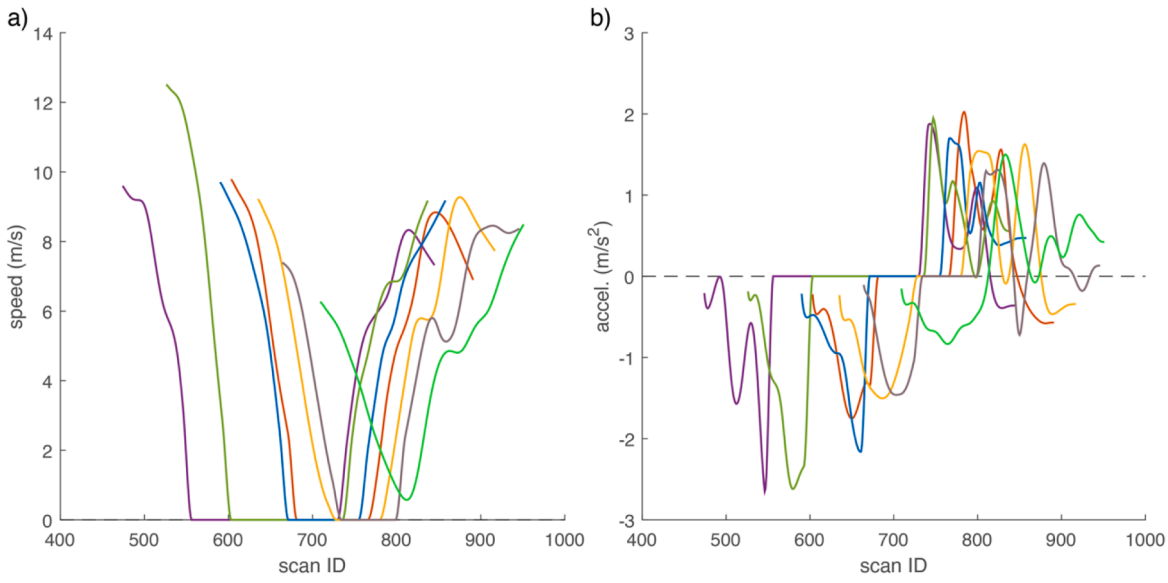

Fig. 9. (a) Speed and (b) acceleration for the seven vehicles in Platoon X shown in Fig. 8.a.

overtook the bus are subsequently occluded and are not seen again in the FoV. Meanwhile, consider the seven vehicles marked *Platoon X* at scan 800 in Fig. 8.a, Fig. 9 shows the corresponding speed and acceleration data for these vehicles as they stop for the pedestrian signal and then discharge.

4. Conclusions

This paper developed a non-model based vehicle tracking methodology for extracting road user trajectories as they pass through the field of view of a 3D LiDAR sensor mounted on the side of the road. To minimize the errors, our work breaks from conventional practice and postpones target segmentation until after collecting LiDAR returns over many scans. Specifically, our method excludes all non-vehicle returns in each scan and retains the ungrouped vehicle returns. These vehicle returns are stored over time in a spatio-temporal stack (ST stack) and we develop a vehicle motion estimation framework to cluster the returns from the ST stack into distinct vehicles and extract their trajectories. This work uses a Volumetric Neighborhood Clustering to segment distinct tracks in the ST stack into distinct clusters. The clusters are then processed to resolve overgrouping and address dropouts due to occlusion. This processing includes removing the impacts of the target's changing orientation relative to the LiDAR sensor while separately taking care to preserve the crisp transition to/from a stop that would normally be washed out by conventional data smoothing or filtering. The resulting processing shows good performance with real LiDAR data.

This proof of concept study develops the methodology using a single LiDAR sensor, thus, limiting the surveillance region to the effective range of the given sensor. Provided sufficient georeferencing, in practice the surveillance region can be extended indefinitely by deploying multiple LiDAR sensors with overlapping fields of view. On the other hand, we suspect that occlusions would make it difficult to track beyond two lanes, but if that is the case, one could deploy an additional sensor on the opposite side of the road to expand coverage up to 4 lanes or one could place the LiDAR sensor in the median for divided roadways.

The ST stack formation in Section 2.3 and subsequent processing could be made even more robust using a 4D stack: XYZ and time, and then processing the returns in 3D space instead of along a 1D lane. The principles remain the same as one adds dimensions but become much harder to illustrate in 2D figures. Obviously, the processing also becomes more complicated too. Regardless, the reduced dimensionality of this work has demonstrated the viability and effectiveness of the method. Future work will explore higher dimensional processing.

CRediT authorship contribution statement

Yuyi Chang: Conceptualization, Methodology, Software, Formal analysis, Investigation, Visualization, Writing - original draft. **Wen Xiao:** Resources, Data curation, Writing – review & editing, Funding acquisition. **Benjamin Coifman:** Conceptualization, Methodology, Formal analysis, Investigation, Visualization, Supervision, Funding acquisition, Writing - original draft.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This material is based in part upon work supported in part by the National Science Foundation under Grant No. 2023857. The contents of this report reflect the views of the authors who are responsible for the facts and the accuracy of the data presented herein. This report does not constitute a standard, specification or regulation.

This research was also partially supported by the UKCRIC—UK Collaboratorium for Research in Infrastructure & Cities: Newcastle Laboratories (EPSRC award EP/R010102/1) and the National Natural Science Foundation of China (Grant NO. 42201485).

References

Barmpounakis, E., Geroliminis, N., 2020. On the new era of urban traffic monitoring with massive drone data: The pNEUMA large-scale field experiment. Transport. Res. Part C: Emerg. Technol. 111, 50–71.

Bar-Shalom, Y., Daum, F., Huang, J., 2009. The probabilistic data association filter. IEEE Control Syst. Mag. 29, 82–100.

Barth, M., An, F., Norbeck, J., Ross, M., 1996. Modal Emissions Modeling: A Physical Approach. *Transp. Res. Rec.*, Issue 1520 (1996), 81–88.

Cattin, J., Leclercq, L., Pereyron, F., El Faouzi, N.E., 2019. Calibration of Gipps' Car-Following Model for Trucks and the Impacts on Fuel Consumption Estimation. IET Intel. Transport Syst. 13 (2), 367–375.

Chandler, R.E., Herman, R., Montroll, E.W., 1958. Traffic dynamics: Studies in car following. Oper. Res. 6, 165–184.

Chen, D., Laval, J., Zheng, Z., Ahn, S., 2012. A behavioral car-following model that Captures Traffic Oscillations. Transp. Res. B 46 (6), 744–761.

Chen, X., Li, S., Mersch, B., Wiesmann, L., Gall, J., Behley, J., Stachniss, C., 2021. Moving Object Segmentation in 3D LiDAR Data: A Learning-Based Approach Exploiting Sequential Data. IEEE Robot. Autom. Lett. 6, 6529–6536. https://doi.org/10.1109/LRA.2021.3093567.

Chen, C., Luo, J., Parker, K., 1998. Image segmentation via adaptive K-mean clustering and knowledge-based morphological operations with biomedical applications. IEEE Trans. Image Process. 7 (12), 1673–1683.

Cheng, L., Chen, S., Liu, X., Xu, H., Wu, Y., Li, M., Chen, Y., 2018. Registration of Laser Scanning Point Clouds: A Review. Sensors 18, 1641. https://doi.org/10.3390/s18051641.

Coifman, B., Beymer, D., McLauchlan, P., Malik, J., 1998. A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance. Transp. Res. C 6 (4), 271–288.

Coifman, B., Li, L., 2017. A critical evaluation of the next generation simulation (NGSIM) vehicle trajectory dataset. Transp. Res. B Methodol. 105, 362–377.

Coifman, B., Li, L., Sept. 2022. 2022 A New Method for Validating and Generating Vehicle Trajectories from Stationary Video Cameras. IEEE Trans. Intell. Transp. Syst. 23 (9), 16227–16236.
Coifman, B., McCord, M., Mishalani, R.G., Iswalt, M., Ji, Y., 2006. Roadway traffic monitoring from an unmanned aerial vehicle. IEE Proceedings-Intelligent Transport Systems 153 (1), 11–20.
Coifman, B., Wu, M., Redmill, K., Thornton, D.A., 2016. Collecting ambient vehicle trajectories from an instrumented probe vehicle: High quality data for microscopic traffic flow studies. Transportation Research Part C: Emerging Technologies 72, 254–271.
El Sallab, A., Sobh, I., Zidan, M., Zahran, M., Abdelkarim, S., 2018. YOLO4D: A Spatio-temporal Approach for Real-time Multi-object Detection and Classification From LiDAR Point Clouds. Montréal, Canada.
Elhousni, M., Huang, X., 2020. A survey on 3D LiDAR localization for autonomous vehicles, *Proc. 2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, pp. 1879–1884.
Ester, M., Kriegel, H.-P., Sander, J., Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise., *KDD*. pp. 226–231.
Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24, 381–395.
Gao, B., Coifman, B., 2007. In: A Vehicle Detection and Tracking Approach Using Probe Vehicle LIDAR Data. Springer, pp. 675–685.
Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M., 2020. Deep learning for 3D point clouds: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence.
Jiang, R., Hu, M., Zhang, H.M., Gao, Z., Jia, B., Wu, Q., 2015. On Some Experimental Features of Car-Following Behavior and how to Model Them. Transp. Res. B Methodol. 80, 338–354.
Kovvali, V., Alexiadis, V., Zhang, L., 2007. Video-Based Vehicle Trajectory Data Collection, *Proc*. Of the 86th Annual TRB Meeting.
Krajewski, R., Bock, J., Kloeker, L., Eckstein, L., 2018. The HighD dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems, *Proc*. In: *21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 2118–2125.
Kuppili, S.K., Alshetty, V.D., Diya, M., Nagendra, S.S., Ramadurai, G., Ramesh, A., Gulia, S., Namdeo, A., Maji, K., Bell, M., Goodman, P., Hayes, E., Barnes, J., Longhurst, J., De Vito, L., 2021. Characteristics of real-world gaseous exhaust emissions from cars in heterogeneous traffic conditions. Transp. Res. Part D: Transp. Environ. 95, 102855.
Lee, H., Coifman, B., 2015. Using LIDAR to Validate the Performance of Vehicle Classification Stations. J. Intell. Transp. Syst. 19 (4), 355–369.
Li, Y., Ibanez-Guzman, J., 2020. Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems. IEEE Signal Process Mag. 37, 50–61. https://doi.org/10.1109/MSP.2020.2973615.
Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017. Pointnet: Deep learning on point sets for 3D classification and segmentation. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652-660. 2017.
Ramasamy, S., Sabatini, R., Gardi, A., Liu, J., 2016. LIDAR obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid. Aerosp. Sci. Technol. 55, 344–358. https://doi.org/10.1016/j.ast.2016.05.020.
Rauch, H.E., Tung, F., Striebel, C.T., 1965. Maximum likelihood estimates of linear dynamic systems. AIAA J. 3, 1445–1450.
Sazara, C., Nezafat, R.V., Cetin, M., 2017. Offline Reconstruction of Missing Vehicle Trajectory Data from 3D LIDAR. *Proc. of the IEEE Intelligent Vehicle Symposium*, Redondo Beach CA, June 11-14, 2017.
Strigel, E., Meissner, D., Seeliger, F., Wilking, B., Dietmayer, K., 2014. The ko-per intersection laserscanner and video dataset, *Proc*. In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 1900–1901.
Tarko, A.P., Ariyur, K.B., Romero, M.A., Bandaru, V.K., Lizarazo, C.G., 2016. *TScan: Stationary LiDAR for traffic and safety studies—object detection and tracking*. Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2016/24. West Lafayette, Purdue University, IN.
Thornton, D., Redmill, K., Coifman, B., 2014. Automated Parking Surveys from a LIDAR Equipped Vehicle. Transport. Res.- Part C 39, 23–35.
Torr, P. H. S., and A. Zisserman, 2000. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. Computer Vision and Image Understanding.
VRU Trajectory Dataset, 2022, https://www.th-ab.de/vru-trajectory-dataset.
Tsanakas, N., Ekström, J., Olstam, J., 2022. Generating Virtual Vehicle Trajectories for the Estimation of Emissions and Fuel Consumption. *Transportation Research Part C: Emerging Technologies*, Volume 138, Issue 0, 2022, 103615.
Wu, M., Coifman, B., 2019. Quantifying what goes unseen in instrumented and autonomous vehicle perception sensor data– a case study. Transportation Research Part C: Emerging Technologies 107, 105–119.
Zhang, J., Xiao, W., Coifman, B., Mills, J.P., 2020. Vehicle tracking and speed estimation from roadside lidar. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 13, 5597–5608.
Zhao, J., Xu, H., Liu, H., Wu, J., Zheng, Y., Wu, D., 2019. Detection and tracking of pedestrians and vehicles using roadside LiDAR sensors. Transport. Res. Part C: Emerg. Technol. 100, 68–87.