Robust Vehicle Tracking for Urban Traffic Videos at Intersections

Li C.¹ ², Chiang A.¹ ², Dobler G.¹ ³, Wang Y.², Xie K.⁴, Ozbay K.⁴, Ghandehari M.⁴, Zhou J.¹, Wang D.¹

¹Center for Urban Science + Progress, New York University, NY, USA
²Department of Electrical and Computer Engineering, New York University, NY, USA
³Department of Physics, New York University, NY, USA
⁴Department of Civil and Urban Engineering, New York University, NY, USA

chenge.li, atc327, greg.dobler, yw523, kun.xie, kaan.ozbay, masoud, jz2308, dw1609 @nyu.edu

Abstract

We develop a robust, unsupervised vehicle tracking system for videos of very congested road intersections in urban environments. Raw tracklets from the standard Kanade-Lucas-Tomasi tracking algorithm are treated as sample points and grouped to form different vehicle candidates. Each tracklet is described by multiple features including position, velocity, and a foreground score derived from robust PCA background subtraction. By considering each tracklet as a node in a graph, we build the adjacency matrix for the graph based on the feature similarity between the tracklets and group these tracklets using spectral embedding and Dirichelet Process Gaussian Mixture Models. The proposed system yields excellent performance for traffic videos captured in urban environments and highways.

1. Introduction and Motivation

Traffic safety is one of the most pressing modern, urban problems. In 2014 alone, there were 6,064,000 traffic accidents in the United States resulting in 29,989 civilian deaths [1]. Typically traffic conflicts are characterized and cataloged by subjective observations in the field or by reviewing field videos manually, resulting in sparse classification of potential hazards [2]. Furthermore, the identification of high-risk locations and the assessment of safety improvement solutions are done through the use of historical crash data, implying that traffic safety engineers must wait to observe actual accidents. Given rapid advances in object tracking methodology in recent years, computer vision (CV) techniques for urban traffic surveillance offer a promising alternative to develop non-crash, surrogate safety measures [3]. A number of recent studies have used CV-based methods for traffic conflict analysis [4, 5, 6, 7, 8, 9].

In this paper, we develop a robust system to automatically extract vehicle trajectory data from video data obtained by existing traffic cameras from the New York City Department of Transportation (NYCDOT). These trajectory information, which will be used in our subsequent study, can be used to develop realistic surrogate safety measures to both identify high risk locations and assess implemented safety improvements on time scales of days to weeks rather than years.

1.1. Background and Related Works

Although there has been a substantial research effort addressing vehicle tracking, designing accurate methods in the real world (especially in busy urban intersections) remains a significant challenge. Automated traffic analysis systems typically fall into two categories. The first
involves tracking of whole objects identified by foreground/background separation \[4\] and vehicle detection \[5, 6\]. This method is less robust to heavy congestion and occlusion effects. The second tracks local features on objects (vehicles) which are subsequently grouped together to form coherent trajectories of individual objects (e.g., \[7, 8\]). This results in additional computational complexity and sensitivity to noise, but the feature points based methods handle occlusions more effectively and are more robust to varying lighting conditions and diverse vehicle motion patterns. \[7\] implement feature based tracking by grouping features according to a common motion constraint (i.e., constant feature point separation over time as they move with little relative motion). N. Saunier et al in \[8\] extend this by connecting (or disconnecting) Kanade-Lucas-Tomasi (KLT) feature point tracklets based on grouping and segmentation thresholds in space. These approaches, which attempt to capture common motion by simply thresholding the tracklet pairs’ spatial and temporal relationships, often fail in dense urban traffic videos. Proper thresholds need to be adaptive for different videos or even vehicles of different sizes in the same video. Failure to sufficiently adapt these parameters to changing conditions strongly affects the ultimate performance of the tracking. \[9\] recently combined both background subtraction and low-level feature identification by linking the foreground “blobs” to form tracklets by matching low-level FREAK (Fast retina keypoint) \[10\] feature points within each blob. A Finite State Machine (FSM) was used to model different situations such as fragmentation, splitting, and merging of the blob tracklets to form the final tracks. The diversity of traffic videos results in complex FSM transitions and the performance in different scenarios depends strongly on factors including the duration of “missing” an object and its fragmentation factor, both of which are difficult to optimize.

Our system is based on the feature point tracking approach, but with a unique methodology designed for use in street-level traffic camera video. As we describe below, we first use a KLT tracker to track feature points and generate tracklets across video snippets. Each vehicle can have many feature points and thus many tracklets. We overcome complexities due to the presence of false detections (e.g., pedestrians, shadows, parked cars) by automatically clustering the tracklets into individual vehicles based on their spatio-temporal features as well as higher-level features derived from foreground/background separation. Instead of grouping by thresholding using multiple criteria, we integrate both low-level and high-level information into an adjacency matrix and used spectral clustering and Dirichlet Process Gaussian Mixture Models (DPGMM) to group and segment all tracklets automatically.

1.2. Challenges in Urban Traffic Video From Street-Level Cameras

Successful systems for the detection and classification of vehicles on highways – which are typically characterized by homogeneous and constant traffic flows with significant vehicle separations have been developed. In \[6\], Kim et al used the 3D-model-based vehicle detection algorithm on the NGSIM dataset \[22\] to achieve 85% accuracy. However, these algorithms require accurate models for many different types of vehicles, and are difficult to apply to urban environments where vehicles have deformable appearances due to occlusions or viewing angles. Increased congestion, shadowing, occlusion, and pedestrians present significant challenges in urban traffic video. Furthermore, in the case of our NYCDOT cameras, the videos themselves have low resolution (480 x 640 pixels), frequent illumination changes among frames due to the camera’s auto-iris feature, and vibrate due to wind.

As shown in Figure 2, the cameras are mounted at heights of several meters (as opposed to the high bird’s eye view that is typical in highway datasets) leading to a high degree of partial occlusions in dense traffic. These dense traffic conditions also lead to stop-and-go conditions, complicating classical learned background techniques (e.g.,...
Mixture of Gaussians, non-parametric models, etc.) as the stopped vehicles are hard to separate from the background. Shadowing by the vehicles themselves also presents complications as they generate feature points that are difficult to distinguish from those on actual vehicles and these shadows tend to connect adjacent vehicles.

2. Proposed System

Figure 1 illustrates the components of our methodology. The complete system integrates object-level and feature-level information into coherent vehicle trajectories that can then be used to develop surrogate safety measures for a given intersection.

2.1. KLT Feature Point Tracker

We first implement a standard Kanade-Lucas-Tomasi (KLT) tracker [11] to generate short tracklets that typically track corners on vehicles. We used KLT tracker because it is fast, flexible with many tunable parameters. We are using the off-the-shelf implementation from openCV library. As we show in Figure 3, a given vehicle can produce more than one tracklet and, given that KLT points come and go due to noise, low spatial resolution, poor lighting conditions, or occlusions, these tracklets need not overlap in time. That is, the KLT tracker may lose track of a feature point at some frame, and then detect and track that point again in a future frame, and so those tracklets corresponding to a given vehicle typically live in different time intervals and have short lengths. Thus, each tracklet $i$ consists of positions $\vec{x}_i(t_j)$ and velocities $\vec{v}_i(t_j)$ for frames $t_j$ in which the KLT point is detected.

2.2. Tracklet Filtering and Smoothing

The KLT tracker described above will also produce “spurious” tracklets for stationary points (e.g., corners on buildings or crosswalk stripes) or very short tracklets due to noise fluctuations. If we denote the maximum velocity of a tracklet by $v_{\text{max}}$ and the total number of frames by $N_{\text{fr}}$, we filter out these spurious detections by imposing minimum $v_{\text{max}}$ and $N_{\text{fr}}$ thresholds. By applying the threshold to $v_{\text{max}}$ (as opposed to, for example, the average velocity magnitude $\langle |\vec{v}| \rangle$) we ensure that stopped cars which subsequently move are not removed from our tracklets sample.

To reduce noise in the remaining tracklets due to camera shaking, lighting changes, and pixelization, we smooth the trajectories both spatially and temporally. The KLT feature point locations have sub-pixel precision, and these locations also varies with the camera shakeness. Spatial smoothing is done to get rid of these flickering small numbers by fitting the $Y(X)$ curve with piece-wise linear first order splines in the $(x, y)$ plane. Temporal smoothing is performed on $\vec{x}(t)$ across frames with the LOWESS filter [12]. The smoothed and filtered results are shown in Figure 4.

2.3. Robust PCA Background Subtraction

Once tracklets are derived for a video snippet, we wish to group them according to the individual vehicles to which they belong. Simply thresholding the trajectory pairs’ spatial and temporal relationships often fails in dense urban traffic video. For example, a tracklet representing the rear bumper of a leading vehicle may be quite close to one representing the front bumper of a trailing vehicle in $(\vec{x}, \vec{v})$ space. Temporal smoothing is performed on $\vec{x}(t)$ across frames with the LOWESS filter [12]. The smoothed and filtered results are shown in Figure 4.
Figure 6. Four corresponding point pairs are shown for the street-level (from the NYCDOT camera; top) and overhead (from Google Earth; bottom) perspectives of the same scene. Using this information we are able to construct the homography matrix and map street-level trajectories into parallel tracklets (see Figure 7).

separation of the video. Tracklets belonging to the same vehicle will ideally fall into the same foreground blob. Those which do not fall in any blob are assigned to the nearest blob (defined by distance to a blob edge). This blob label represents a high-level feature as opposed to a local, low-level feature and will be used to compare tracklets when clustering tracklets (see §2.5).

To generate these foreground blobs we use Robust PCA (rPCA) [13], which decomposes the video data into a low rank background and a sparse foreground and outperforms simple foreground/background separation methods especially in the presence of frequent illumination changes. The rPCA is implemented via incremental Principle Component Pursuit [14, 15], which is computationally more efficient than the original rPCA method. Despite its efficacy, as shown in Figure 5, for congested urban scenes rPCA often forms either incomplete, unconnected blobs (multiple blobs for a single vehicle, which we mitigate via binary closing) or overconnected blobs (a single blob for multiple vehicles). Furthermore, as with other foreground/background separation techniques, stationary vehicles are grouped into the low rank background. This over- and under-segmenting of blobs complicates blob label comparisons, so in the clustering step below we use the blob index to assign each tracklet i a blob “position” defined as its center of mass $\vec{b}_{cen,i}$.

Figure 7. Unwarped (left) and warped (using the information from Figure 6; right) trajectories from the street-level NYCDOT cameras.

2.4. Perspective transformation

The NYCDOT cameras are typically only several meters above ground level. The result is that vehicles have significantly different pixels sizes based on their location within a frame, and tracklets of different points on the same vehicle are not parallel on the image plane, as shown in Figure 3). Thus, tracklet clustering algorithms that depend on $\vec{x}$ and $\vec{v}$ in pixel units will perform poorly.

The alternative is to either use scene information to convert trajectories to the world space or to perform perspective warping to parallelize the trajectories. Since we are not concerned with absolute velocities and positions, and since we do not know the intrinsic camera parameters of the NYCDOT cameras, we implement the latter approach by identifying four corresponding point pairs in a NYCDOT frame and a Google Earth map for the same intersection to calculate the homography matrix (see Figure 6). An example of the rectified trajectories is shown in Figure 7.

2.5. Grouping Tracklets Using Spectral Clustering

Spectral clustering has previously been used to analyze traffic at intersections by clustering vehicle trajectories in video. For example, [16, 17, 18] identify global patterns of an intersection by identifying clusters which represent particular characteristic trajectories; i.e., a cluster might represent a left or right turn, on-coming traffic, etc. Since their primary objective was to assess the global pattern (as opposed to vehicle-vehicle interactions as is our goal), the assumption is that the individual trajectory information is available after manual rectification or that per-vehicle trajectories were not needed.

We group the $N_{tr}$ tracklets into coherent vehicle trajectories via spectral clustering [19] as follows. For each pair of trajectories $i$ and $j$ that overlap temporally, we define an adjacency based on four features: the maximum positional separation along the tracklet $x_{ij,max} \equiv \max_k |\vec{x}_i(t_k) - \vec{x}_j(t_k)|$, the maximum velocity separation in two dimensions along the tracklet $v_{x,ij,max} \equiv \max_k |v_{x,i}(t_k) -$


\( v_{x,j}(t_k) \) and \( v_{y,i,j,\text{max}} \equiv \max_k |v_{y,i}(t_k) - v_{y,j}(t_k)| \), and the maximum separation of the center of mass of the blob labels \( b_{\text{cen},i,j,\text{max}} \equiv \max_k |b_{\text{cen},i}(t_k) - b_{\text{cen},j}(t_k)| \). The adjacency between two tracklets \( A_{ij} \) is defined as,

\[
\ln A_{ij} = -\vec{w} \cdot \vec{f}_{ij},
\]

where \( \vec{w} \) is a weight vector for each feature in \( \vec{f}_{ij} \equiv (x_{ij,\text{max}}, v_{x,i,j,\text{max}}, v_{y,i,j,\text{max}}, b_{\text{cen},i,j,\text{max}}) \).

The exponential kernel resulted in more accurate tracks in our case than the more commonly used Gaussian kernel, and so the final result is based on the exponential kernel construction. The weight vector represents hyperparameters which must be tuned to give optimal results for a given video. Furthermore, we note that we use the maximum values for position and velocity separation instead of mean or median values since, for “same-vehicle tracklet pairs” that correspond to the same rigid vehicle, the mean and maximum separation remain similar throughout the vehicle’s trajectory, whereas for “different-vehicle tracklet pairs” the maximum can be very different, making it a stronger discriminant (for example two vehicles traveling side-by-side for the majority of the video and then diverging towards the end).

We note that, if \( i \) and \( j \) fall in the same blob for all \( k \), \( b_{\text{cen},i,j,\text{max}} = 0 \). This representation using center of mass is more robust than just the blob index representation because even if the vehicle is over-segmented into multiple blobs, the center locations are still relatively near each other. However, in the case of a large blob that connects multiple vehicles, the associated tracklets must rely on other features to further distinguish them. Lastly, for tracklets which do not overlap at any frame \( t_k \), we set \( A_{ij} \) to zero.

We use a modified version of the implementation of spectral clustering in the SciKit-Learn (SKL) Python package [20] to individually cluster subcomponents of the full \( N_{\text{tr}} \times N_{\text{tr}} \) adjacency matrix. The subcomponents are defined by thresholding \( A_{ij} \) according to \( x_{ij,\text{max}} \leq d \) to form connected components which are never separated by more than a distance \( d \). This drastically reduces the computation time since (while not truly sparse) \( A_{ij} \) is dominated by very small adjacency values. For each connected component, the SKL spectral clustering implementation performs the spectral embedding and dimensionality reduction according to the \( K \) lowest eigenvalues of the Laplacian matrix. In our modified implementation, rather than using the SKL default clustering methods (K-Means or discretization, both of which require a known number of clusters which is not our case here) on this lower dimensional manifold, we use a Dirichlet Process Gaussian Mixture Model (DPGMM) [21] to identify the number of clusters and label each tracklet.

Lastly, it is clear that for long videos \( N_{\text{tr}} \) can be very large making the spectral clustering of \( A_{ij} \) computationally intractable. To get around this constraint, we segment videos into snippets of ~ 600 frames (corresponding to roughly 2 minutes of NYCDOT video)\(^1\) and cluster within each snippet. The relatively few tracklets which overlap two snippets are propagated into the subsequent adjacency matrix thus unifying labels across snippets. An example of the final clustered tracks is shown in Figure 8.

3. Experimental Results and Evaluation

We evaluate our system performance on two different datasets: NYCDOT video recorded at Canal St & Baxter St in Manhattan’s Chinatown and the NGSIM US-101 highway dataset [22]. Both have dense traffic flows, but differ in their camera orientation. The NYCDOT video is a street-level camera with many occlusions, speed variations, and perspective effects, while the NGSIM video has a bird’s eye view with mostly parallel trajectories of uniform velocity.

Ground truth trajectories consisting of vehicle center and bounding box for each frame were manually generated for

\(^1\)We derive KLT tracklets on raw 30 fps video for accurate tracking and then down sample by a factor of 6 prior to any subsequent steps to reduce computational complexity.
For each dataset with custom software, see Figure 9. To compare this groundtruth with automatically derived trajectories, we use two evaluation metrics proposed by [23], miss rate and over-segmentation rate. Miss rate $M$ is defined as the ratio of the number of missed vehicles to the total number of moving vehicles, and over-segmentation rate $O$ as the number of vehicles that are represented by more than one trajectory over the total number of vehicles.

For each ground truth trajectory, we compute the distance of this trajectory and all automatically generated trajectories. We only consider successful tracking to be those cases for which the automatically generated trajectory is within the manually generated bounding box for over 30% of its temporal duration. When a ground truth vehicle has more than one automatically generated trajectories as successful trackings, we consider this as an over-segmentation case, see Figure 10 and 11.

We find that our tracker is quite successful on the street-level NYCDOT video with the miss rate $M = 21\%$ despite the substantial difficulties associated with that scene. A simple thresholding of the adjacency matrix yields a substantially higher miss rate $M = 53\%$. For the NGSIM data set our method also performs quite well with miss rate $M = 18\%$, slightly worse than $M = 15\%$ in [6], though we point out that the details of our tracker were designed for optimal performance on street-level video, not overhead video with nearly parallel, nearly uniform velocity intrinsic trajectories.

For both datasets, the over-segmenting cases can occur. Specifically we have an over-segmentation rate $O = 31\%$ for NGSIM dataset. Over-segmented trajectories can be merged though a post-processing or clustering, which will substantially decrease the over-segmenting rate. It is also important to note that the ground truth trajectories are located at the centroids of the vehicles. Since the positions of our final trajectory locations are defined as the median of the clustered tracklets at each frame, they may deviate from the vehicle centroids due to the incomplete KLT feature point coverage over the vehicle or even missed feature points. The final clustered result can also be smoothed through post-processing, which we will leave as future work.

4. Conclusion and Future Work

In this paper we have developed an automatic system for vehicle tracking designed for street-level video in urban environments. Our method uses a combination of high-level and low-level vehicle features to group feature point tracklets into coherent trajectories via spectral clustering. We have shown that this system can also be effectively applied to bird’s eye view highway traffic video. Given the robustness of our method, even in very challenging video with multiple occlusions, variable velocity, and a complex background scene, the derived trajectories can be used to develop surrogate safety metrics for post-facto traffic analysis.
Acknowledgment

This work was funded through a sponsored research grant from the American International Group. Dr Gregory Dobler was partially supported by a Complex Systems Scholar grant from the James S. McDonnell Foundation.

References


