Automatic Parsing of Lane and Road Boundaries in Challenging Traffic Scenes

Mohamed A. Helala, Faisal Z. Qureshi, Ken Q. Pu
University Of Ontario Institute of Technology, Faculty of Science, 2000 Simcoe Street North, Oshawa, Ontario, Canada, L1H 7K4

Abstract.
Automatic detection of road boundaries in traffic surveillance imagery can greatly help subsequent traffic analysis tasks, such as vehicle flow, erratic driving, stranded vehicles, etc. This paper develops an online technique for identifying the dominant road boundary in video sequences captured by traffic cameras under challenging environmental and lighting conditions, e.g., unlit highways captured at night. The proposed method works at real-time of upto 20 frames per second, and generates a ranked list of road regions that identify road and lane boundaries. Our method begins by segmenting each frame into a set of superpixels. An adaptive sampling step approximates superpixel contours to a collection of edge segments. Next, we show how online hierarchical clustering can be efficiently used to organize edges into clusters of co-linearly similar sets. Promising clusters are paired with each other to form cluster pairs. Then, we present and prove a statistical ranking measure that is used along with road-activity and perspective cues to find the dominant road boundaries. We evaluate the proposed approach on two real world datasets to test our method under camera viewpoint changes and extreme environmental and lighting conditions. Results show that our method outperforms two state-of-the-art techniques in precision, recall and runtime.

Keywords: Traffic Video Analysis, Video Surveillance, Hierarchical Clustering, Statistical Ranking, Superpixel Segmentation, Road and Lane Detection..

Address all correspondence to: Mohamed A. Helala, University Of Ontario Institute of Technology, Faculty of Science, 2000 Simcoe Street North, Oshawa, Ontario, Canada, L1H 7K4; Tel: +1 905-721-8668 ext: 2387; Fax: +1 905-721-3304; E-mail: mohamed.helala@uoit.ca

1 Introduction

Roadway traffic surveillance cameras play a major role in modern traffic management systems [1, 2]. Imagery captured by traffic cameras can be used, among other things, for traffic flow monitoring and analysis, detecting and responding to traffic accidents, automatic highway toll billing, etc. Increasingly, videos recorded by traffic cameras are available in real-time to the general public on their hand-held devices, such as smart phones. TrafficLand[*] website, for example, provides live streams of over 18 thousand traffic cameras from more than 200 North American cities. Access to live feeds from traffic cameras enable commuters to pick an alternate route, say to avoid traffic congestion. It is then not surprising that traffic cameras are being installed in increasing numbers on roads and highways in and around big urban centers. We can only conclude that the trend for using traffic cameras to monitor roads and highways will continue and that the number of traffic cameras will continue to increase. Corollary to this statement is that the data collected by smart cameras will also experience unprecedented growth. Using humans to monitor the video data collected by these cameras is prohibitively costly, and we need automated techniques for consuming and analyzing this data.

This work focuses on the task of road boundary detection in traffic cameras. The ability to identify road regions simplifies subsequent traffic analysis tasks. Traffic cameras are typically mounted

[*]TrafficLand: http://www.trafficland.com/ (*last accessed on 10 May 2015*).

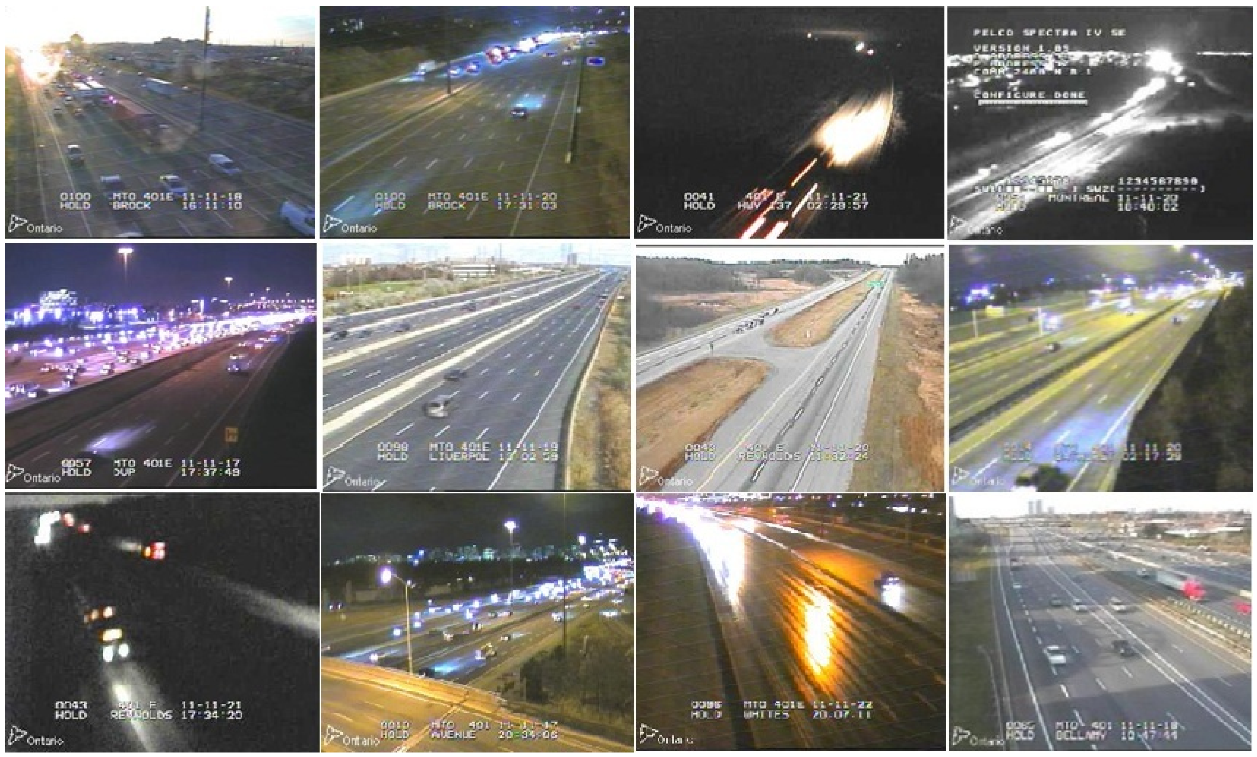


Fig 1: Challenging environmental conditions encountered by traffic surveillance cameras.

on polls along the highways. These cameras are almost never calibrated, since it is tedious and extremely costly to maintain the calibration over the course of the lifetime of a traffic camera. These are also often pan/tilt/zoom cameras, which allows an operator to change the viewing direction of a camera if needed. Therefore, it is desirable to have a road boundary detection algorithm that requires minimal human intervention. Traffic cameras are mounted outdoors and images captured by these cameras exhibit a wide variety of lighting and environmental conditions. For example, these cameras can experience significant sway under strong winds, rendering classical background subtraction infeasible. Roads are barely visible at nighttime or during weather conditions, such as snow, fog, and heavy rain, that can lead to low-visibility. Strong sunlight can give rise to strong, exaggerated shadows. Wet roads when illuminated by vehicle headlights give rise to strong specular reflections. Figure 1 shows a collection of images gathered from traffic cameras mounted along highway 401 in Ontario, Canada. The road detection method presented here is designed to deal with the large environmental and lighting conditions observed in traffic cameras.

Broadly speaking, the existing methods for road and lane detection can be classified into three categories: (1) activity-driven [3, 4]; (2) feature-driven [5, 6, 7]; and (3) model-driven [8, 9, 10]. The activity-driven approaches divide the traffic scene into active (road) and inactive (non-road) regions and construct an activity map based on the observed vehicular motion in the scene. The feature-driven approaches extract image features and organize them into meaningful structures to detect road and lane boundaries. The model-driven approaches casts road detection as a classification task or as model fitting. [10], for example, uses convolutional neural network to classify image regions into road and non-road sections. Given a traffic image, this technique generates a confidence map that assigns each pixel a likelihood of belonging to road regions. [8, 9], on the other hand, attempt to fit a geometric road model to the traffic scene to identify road regions.

This paper focuses on the automatic detection of dominant road boundaries in videos collected by traffic cameras. For our purposes, dominant road boundary divide the image into road and non-road regions. Dominant road boundary is particularly suited for highway traffic cameras. The detected regions can be used to support more efficient higher-level analysis—such as examining traffic flows, detecting erratic driving behavior, identifying stranded vehicles, etc.—since subsequent analysis can focus only on road regions. The proposed approach works as follows. First, it collects low-level edge-like features from a sequence of images (feature-driven). These features are then clustered using hierarchical clustering. Each cluster represents a potential road boundary. We model road boundaries as straight lines (model-driven). The clusters are then ranked using χ^2 and Student t statistical measures. The top-ranked clusters are combined to form road boundary pairs. Each pair is then ranked using vehicle activity (activity-driven) and perspective cues. The top ranked pair is chosen as the dominant road boundary.

The method outlined herein extends our work in [11]. Unlike [11], the current method uses online hierarchical clustering. Online hierarchical clustering allows our method to maintain dominant road boundary even as traffic cameras are re-positioned or their view directions are changed. Previously, one had to reinitialize the system once the cameras are moved. We attempted to address this problem by re-initializing the system every 25 frames, where the number 25 was chosen by trial and error. The system needed to regenerate a new cluster hierarchy every 25 frames, which resulted in a performance hit. In contrast to our previous work, here we use ClusTree [12], an online top-down hierarchical clustering algorithm, which maintains (approximate bottom-up) hierarchical clustering (tree) over a sliding window; new evidence is added to the tree and the stale information is removed from it. This not only results in significant speed up over our previous method that did not use online hierarchical clustering. It also enables our system to adaptively maintain road boundaries as cameras are moved. Live vision systems, such as a traffic surveillance setup, that continuously process incoming imagery are an instance of vision stream processing pipelines. Within the context of (visual) stream processing [13], hierarchical clustering used in [11] acts as a *blocking operator*. Blocking operators adversely effect the runtime performance and scalability of stream processing; therefore, blocking operators are undesirable in live systems. The online hierarchical clustering algorithm used here, on the other hand, is a *non-blocking operation*. This is indeed the primary motivation for using online hierarchical clustering for our purposes. The results bear that the current system outperforms several state-of-the-art systems, including our prior work, on two challenging datasets.

We evaluate the proposed method on two real-world datasets collected from the highway 401 traffic cameras. The first dataset comprises 14 traffic videos exhibiting a variety of environmental and lighting conditions. Each video consists of 25 low-resolution daytime and 25 low-resolution nighttime images. This dataset was also used in [11]. The second dataset consists of 1627 frames long traffic video, showing camera viewing direction switching. We compare our approach against three other schemes: Gabor filter based method that appeared in [7]; a recent approach that uses convolutional neural networks [10]; and the last iteration of our method [11]. Our method outperforms both Gabor filter approach and convoluational neural networks approach in terms of accuracy and runtime. Plus, the current method is roughly 800 times faster than our previous work in [11].

The contributions of this paper are fourfold. First, we present a novel online method for parsing challenging traffic scenes into the dominant road boundaries. The method locates roads under low-visibility and difficult lighting conditions. Second, our method can process low-resolution (320×240) videos at rates of upto 20 frames per second on 2.9 GHz quad-core AMD Athlon processor. This makes it useful for processing live video streams and handling the bandwidth limits of large camera networks. Our method is able to maintain road boundary even when the cameras are re-positioned or their viewings direction changed. Finally, we develop a statistical measure for ranking the road edge clusters that eschews prior knowledge or any heuristics. We believe that this statistical measure may prove useful for ranking clusters in other situations as well.

2 Related Work

As stated earlier, the techniques of lane and road detection can be broadly classified into three categories: (1) activity-driven, (2) feature-driven and (3) model-driven. Here, we briefly summarize the techniques of each category that are relevant to our method. For a detailed survey, we refer the reader to [1, 14].

Activity-driven Methods: The activity-driven techniques [3, 4, 15] use vehicle motion to build an activity map for the traffic scene, and divides the road region into active (road) and inactive (non-road) regions. The work of Stewart *et al.* [3] developed one of the earliest activity-driven methods. Their method accumulates an activity map that records scene changes resulting from vehicle motion. Then, the traffic scene is divided into either active or inactive areas. Melo *et al.* [4] builds on the idea of [3] and develops a method that incorporates the Kalman filter to track moving vehicles. Then, they model the resulting motion trajectories using second degree polynomials, and apply K-means clustering to calculate lane centers. Recently, Chen *et al.* [15] extends the work of [4] by developing a trajectory similarity distance to improve clustering.

Feature-driven Methods: The feature-driven methods [5, 6, 7] rely on low level image features such as colors and textures to detect the lane and road boundaries. The work of Aly [6] developed a method for detecting lane marks in urban roads. His method applies Lane Analysis using Selective Regions (LASeR), which requires camera calibration and uses inverse perspective mapping to construct a top-view of the road. Next, the algorithm applies image filtering and thresholding to extract lane features. Finally, the RANSAC algorithm is used to ignore feature outliers and fit Bezier Splines to lane boundaries. Satzoda *et al.* [5] proposed a similar method to [6] for detecting lanes. However, their work processes selected image bands and applies steerable filters to extract lane features. They also uses a lane geometric model to deal with feature outliers. Kong *et al.* [7] developed an alternative method that divides the road detection problem into two task: 1) estimation of the road vanishing point, 2) segmentation of the road region based on the estimated vanishing point. The estimation task applies Gabor filters to extract texture orientations, and feeds them into a soft voting scheme to estimate the vanishing point. Then, the segmentation task uses the detected vanishing point as a constraint to identify the dominant road boundary.

Model-driven Methods: The model-driven methods [8, 9, 10, 16] performs either road classification or model fitting. Road classification aims to learn a prior model for road regions, which is used later to assign each pixel, a likelihood of belonging to road regions. For example, Brust *et al.* [10] presented an algorithm that uses convolutional neural networks to classify image patches of belonging to either road or non-road regions. The algorithm learns a prior model that incorporates both spatial and appearance information of image patches belonging to road regions. Then, the neural network generates a classification map that assigns the likelihood of each pixel. Model

Algorithm 1 Overview of the propose algorithm for finding top-K dominant road and lane regions.

```
input: Image sequence
output: Dominant road boundary, K
1: Divide each image into homogeneous regions through superpixel segmentation.
2: Approximate each superpixel contour with polygons to get edges.
3: Perform online hierarchical clustering on these edges.
4: Use statistical measures ($\chi^2$ and Student-t test) to identify the top-ranked clusters, each cluster
   represents a road boundary in the image.
5: Construct top-ranked cluster pairs through perspective filtering and road-activity analysis.
6: Return the top-K ranked cluster pairs as the dominant lane and road boundaries.
```

fitting methods [8, 9, 16], on the other hand, match a geometric road model to the traffic scene. The work of Wang *et al.* [9], for example, proposed a lane detection method based on B-Snakes. This method applies edge detection on the input image and partition it into a number of horizontal segments. Then, the algorithm assumes perspective parallel lane lines, and detects a set of control points along the mid-line of the lane. These points are used to define an active contour model based on B-Splines, and energy minimization is used to deform the contour to both left and right to detect the lane boundary. Zhou *et al.* [8] proposed another lane geometrical model that has four parameters, starting position, lane original orientation, lane width and lane curvature. Their algorithm performs three stages: 1) off-line calibration to estimate the camera parameters, 2) model parameters estimation to locate the lane width and dominant orientation, 3) model matching to find the best lane model. Recently, the method of [16] extends the previous techniques by using geographical information to estimate several road priors. Then, it develops a road generative model that combines the road priors with other contextual cues extracted from the traffic scene, such as horizon lines, lane marks, and vanishing points. The generative model is used to construct a confidence map that assigns each pixel, a likelihood of belonging to road regions.

3 Road Boundary Detection

The proposed technique consists of four steps: 1) line segment detection; 2) online hierarchical clustering; 3) confidence assignment (cluster ranking); and 4) pairwise ranking. We discuss each of these steps below.

3.1 Line Segment Detection

Edges are one of the fundamental features for identifying dominant road and lane boundaries in traffic scenes. However, the appearance changes that result from various environmental, traffic and lighting conditions usually make the output of edge detection techniques very noisy. This requires several filtering steps that may need *a priori* knowledge about the road structure. Another problem in standard edge detection techniques such as Canny edge detection is the tuning of threshold parameters. Such tuning becomes difficult under appearance changes. Nevertheless, a further analysis of traffic scenes shows that road regions usually span large areas of the scenes. So, if we are able to extract a number of abstract regions from a traffic scene, the contours of these regions will be attracted to some or all parts of the lane and road boundaries and form a good candidate set of detected edges.

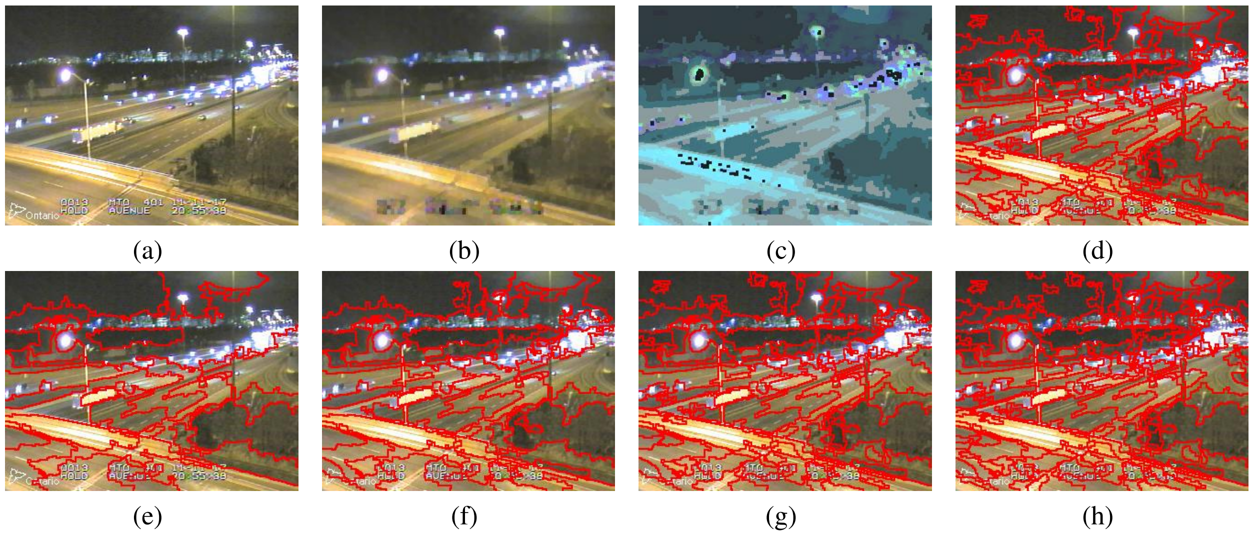


Fig 2: Generating a hierarchy of superpixels by merging neighboring regions based on the hue component. (a) Original image. (b) Image after applying morphological open and close operations. (c) The result of HSV color quantization. (d) Over-segmented Image. (e) No. of superpixels = 25. (f) No. of superpixels = 50. (g) No. of superpixels = 100 and (h) No. of superpixels = 150.

3.1.1 Superpixel Segmentation

We found that superpixel segmentation can provide us with a good set of abstract regions. Superpixel segmentation, originally proposed by Ren and Malik, segments an image into a collection of homogeneous regions where pixels belonging to each region share similar color and texture attributes. Several approaches have been proposed to perform superpixel segmentation [17, 18, 19, 20]. However, these approaches are slow and do not attain the speeds required for traffic scene analysis. Our target is to perform superpixel segmentation at near real-time and at the same time maintain an adequate description for the superpixels. TurboPixels [21] was proposed as a fast algorithm for superpixel segmentation, however, it requires a manual selection of seed points.

We use the superpixel segmentation method of [22], which can generate superpixels at a much higher speed compared to TurboPixels. This technique relies on simple operations to form superpixels and can be implemented in parallel to work at real-time rates. The method starts by applying morphological operations on the input image, followed by Hue-Saturation-Value (HSV) color quantization. Then, adjacent pixels that have similar color values are merged to form initial regions. Neighboring regions are next merged together based on the similarity of the hue color component to form a hierarchy of superpixels.

Figure 2 shows an example of generating superpixels for a nighttime traffic scene. Figure 2b shows the result of applying the morphological open and close operations on the input image. The open operation smooths the contours, breaks narrow strips and removes tiny bumps visible in the image. The close operation, on the other hand, fills gaps, removes small holes and completes contours visible in the image. HSV color quantization (Figure 2c) is then applied by partitioning the HSV color space into $16 \times 8 \times 8$ bins and assigning each pixel, the color of its closest bin. Figure 2d shows the construction of initial regions by merging neighboring pixels having the same color values. Finally, smaller regions are merged with adjacent regions of similar hue values to form a hierarchy of superpixels.

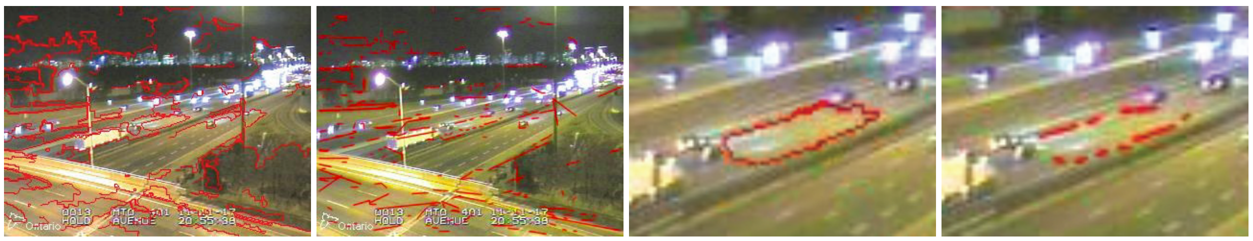

(a) (b) (c) (d)

Fig 3: Polygon approximation using adaptive sampling. (a) Approximated polygons for superpixels in Figure 2e. (b) Same image in (a) but showing edges that are more than 10 pixels in length. (c) Showing a single superpixel from (a). (d) Edges more than 10 pixels for the superpixel in (c).

Algorithm 2 Approximate-Contour.

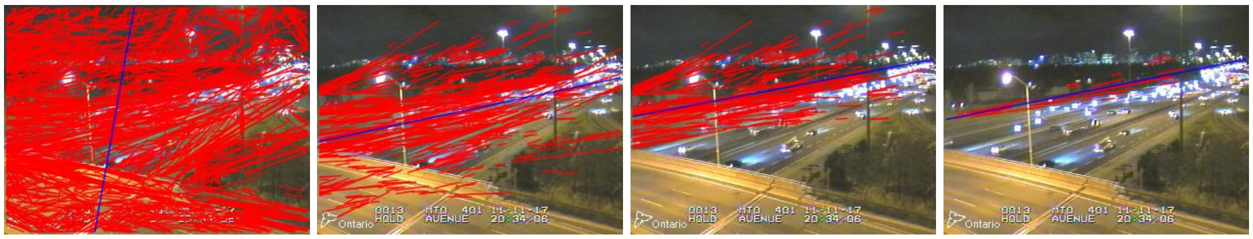
```
input: $\gamma, l, e$
output: Straight Edges.
1: Set $v_l = \gamma(l)$, $v_e = \gamma(e)$, $m = \frac{1}{2}(l+e)$, and $v_m = \gamma(m)$.
2: If the curve's tangents at $v_l, v_m$ and $v_e$ are almost parallel Then,
   Output the straight segment $\overline{v_l v_e}$.
3: Otherwise,
   Approximate-Contour($\gamma, l, m$).
   Approximate-Contour($\gamma, m, e$).
```

Figure 2e to 2h show the segmentation results by varying the number of superpixels from 25 to 150. Notice how the superpixel boundaries stick to the stable edges in the image even when environmental, traffic and lighting conditions affect the size of these superpixels. Our method exploits this characteristic of superpixel segmentation and uses it to collect edge information from each input image.

3.1.2 Contour Approximation

Edges are computed by approximating the superpixel contours as 2D polygons. This approximation is performed using adaptive sampling [23], which is applied independently on the contour of each superpixel. Let $\gamma : [0,1] \rightarrow r^2$ be a parametric curve that represents a superpixel contour lying in 2D space. Adaptive sampling chooses n representative points $0 = t_1 < t_2 < \cdots < t_n = 1$, which defines the vertices $v_1 = \gamma(t_1), \cdots, v_n = \gamma(t_n)$ and best approximates the contour γ while keeping n as small as possible. Initially, each superpixel contour is sampled at two points, a starting point $v_l = \gamma(l)$ where $l = 0$ and an ending point $v_e = \gamma(e)$ where $e = 1$. Then, Algorithm 2 is applied to generate an approximated polygon. This algorithm selects an intermediate point m within the interval $[l, e]$ and tests the co-linearity of the three points v_l, v_m and v_e by evaluating a flatness test. This test ensures that the curve tangents $\overrightarrow{v_l v_m}$ and $\overrightarrow{v_m v_e}$ are almost parallel. If the segment $v_l v_m v_e$ is flat, it will be recorded in the output polygon. Otherwise, the algorithm is recursively called on the intervals $[l, m]$ and $[m, e]$.

Adaptive sampling can sometimes lead to over-sampling, i.e.; a straight segment of the superpixel boundary is divided into much smaller segments. This is handled by deleting the intermediate vertexes. Figure 3 shows an example of approximating the superpixel contours to polygons. Figure 3b shows the generated polygon segments that have a length of more than 10 pixels each.



(a) (b) (c) (d)

Fig 4: The line segments of four sibling clusters in an example clustering hierarchy tree. (a) Root cluster. (b) and (c) Two clusters with large variance. (d) A cluster with small variance and large number of segments which is a good candidate for representing the road boundary.

Figures 3c and 3d zoom onto the contour of one superpixel and show how Algorithm 2 can generate promising edge segments.

3.2 Online Hierarchical Clustering

Our method accumulates a set of approximated line segments from a sequence of frames. Figure 4 presents an example of this set where several concentrations of co-linear segments are visible along the road and lane boundaries. Therefore, finding a good candidate set of clusters that best represents these co-linear segments can allow the algorithm to infer the dominant road boundaries.

In our previous work [11], we applied hierarchical agglomerative clustering with average linkage to construct a hierarchical clustering tree for the accumulated set of segments. Each segment $\mathbf{s}$ is represented in polar coordinates as a 2D vector (ρ, θ). Let $\mathcal{S} = \{\mathbf{s}_i | i = 1 \cdots n\}$ be a set of line segments for a given image sequence. This set is normalized to have zero mean and unit variance. Then, hierarchical agglomerative clustering is initialized by assigning each segment $\mathbf{s}_i$ to a cluster c_i. Next, a cluster hierarchy is generated by recursively merging the closest pairs of clusters as one moves up the hierarchy until we have only one root cluster.

As stated elsewhere, previously our method employed bottom-up hierarchical clustering. This algorithm incrementally builds clustering tree; however, the size of this tree grows as time progresses, i.e., more video frames are processed. We noticed that as a result the system becomes sluggish over time. Furthermore, this system assumes that the location and viewing direction of traffic cameras are fixed, an assumption that does not hold in real life. We addressed these two limitations by partitioning the incoming video stream in chunks. Each chunk was processed separately and we needed to rebuild the clustering hierarchy for every chunk from scratch. We now propose a better solution. The method proposed here uses ClusTree [12]—an online hierarchical clustering. This algorithm builds and maintains the clustering tree by adding new information and removing stale information. Using online hierarchical clustering not only improves the run-time performance of our system. It also enables our method to correctly identify roads even after a camera has been moved.

ClusTree algorithm maintains cluster hierarchy by reacting to changes in existing clusters and creating new clusters when needed. It builds a R-tree based multidimensional index where each node stores a cluster feature vector $CF = (n, LS, SS)$, a timestamp t of last update, and a pointer to its children. Here, n is the number of currently inserted objects, LS is their linear sum and SS is their squared sum. CF represents the sufficient statistics required to calculate and incrementally update the mean and variance of a cluster. In order to maintain up-to-date clustering and forget stale objects, the CF vector is weighted using an exponential decay function $\omega(\Delta t) = \beta^{\lambda \Delta t}$, where λ is the decay rate. This function assigns weights to the feature vector of each cluster as

$$n^{(t)} = \sum_{i=1}^{n} \omega(t - ts_i), \tag{1}$$

$$LS^{(t)} = \sum_{i=1}^{n} \omega(t - ts_i)\mathbf{s}_i, \tag{2}$$

$$LS^{(t)} = \sum_{i=1}^{n} \omega(t - ts_i)\mathbf{s}_i^2, \tag{3}$$

where t is the current time and ts_i is the timestamp at which the line segment $\mathbf{s}_i$ arrived. It was shown by [12] that the additive properties of weighted cluster features are maintained, in addition to temporal multiplicity. For example, a cluster feature $CF^{(t+\Delta t)} = \omega(\Delta t)CF^{(t)}$ if no object is added during the interval $[t, \Delta t]$. Moreover, a cluster with k children has $CF^{(t+\Delta t)} = \omega(\Delta t)\sum_{i=1}^{k} CF_i^{(t)}$, where $CF_i^{(t)}$ is the cluster feature of child i. We refer the kind reader to [12] for the proof of these properties.

When inserting a line segment $\mathbf{s}$ into the clustering tree, the algorithm attaches the current timestamp ts to the object. Then it descends the tree by choosing at each cluster node, the child cluster that has the smallest Euclidean distance between its mean and the inserted object. While descending into a node, the algorithm updates the node's cluster feature by adding the new object, multiplying with the decay function $\omega(t - ts)$ and updating the cluster timestamp $t = ts$. If a child cluster has $n^{(t)} < \beta^{\lambda \Delta t_c}$, it is declared outdated and is deleted. Δt_c is a user defined interval that controls the deletion rate. Descending stops when we reach a leaf node, where the new object is inserted to become a new leaf. Figure 4 shows four sibling clusters in an example clustering hierarchy. Although the root cluster is noisy, the innermost nested cluster can clearly capture one of the road edges.

For each incoming frame, our method updates the clustering tree by inserting the frame's approximated line segments. For each inserted segment, it records the clusters updated by the insertion. Then, it defines the union set of all current updated clusters as the candidate set of frame's dominant lines.

3.3 Confidence Assignment

After generating the candidate set of clusters, the technique ranks them based on the cluster variance and number of samples. This ranking penalizes clusters with high variance or small number of samples. We model the road boundary as a line in the parametric form:

$$\widehat{m}u + \overrightarrow{v}, u \in R, \tag{4}$$

where $\widehat{m}$ is a directional unit vector and $\overrightarrow{v}$ is an offset vector from the origin of a given coordinate system. We can then represent a segment along a certain road boundary line as

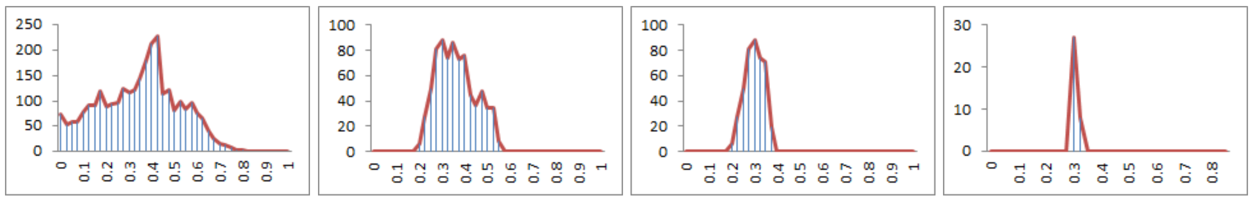
$$\overrightarrow{m_s}u + \overrightarrow{v_s}, u \in [a, b], \tag{5}$$

which can be used to define the following generative model,

$$\overrightarrow{m_s} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \cdot \widehat{m}, \quad \phi \in \mathcal{N}(\mu_1, \sigma_1^2), \tag{6}$$

$$\overrightarrow{v_s} = \rho \overrightarrow{v}, \quad \rho \in \mathcal{N}(\mu_2, \sigma_2^2), \tag{7}$$

where ϕ and ρ are random variables with Gaussian noise that depends on the set of parameters $\theta = (\mu_k, \sigma_k^2)_{k=1,2}$. We can then describe our model by the probability $P(\pi|\theta)$ where the hidden variables $\pi = (\overrightarrow{m}, \overrightarrow{v})$ represent a true road boundary line.


(a) ρ histograms.

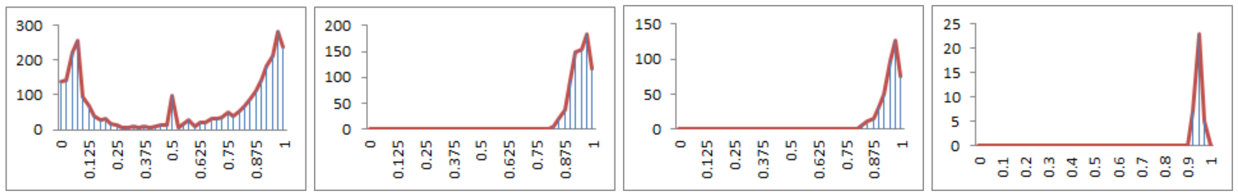

(b) ϕ histograms.

Fig 5: The ρ and ϕ histograms for the four clusters shown in Figure 4a to 4d respectively. (a) shows the ρ histograms and (b) shows the ϕ histograms. Each column corresponds to one cluster. The first column shows noisy histograms for the root cluster, while the second column shows less noisy histograms for the sub-cluster in Figure 4b, with few peaks in the ρ histogram and a normally distributed ϕ histogram. The third column shows a gradually reduced noise for the next sibling cluster and the final column presents the histograms for the road boundary cluster with much reduced Gaussian noise.

Figure 5 shows the ρ and ϕ histograms for the clusters presented in Figure 4. The first column shows the histograms of the root cluster which are very noisy and contain several clear peaks. These peaks indicate the existence of different sibling clusters each with similar ρ and/or ϕ values. The second column shows the histograms of the nested cluster in Figure 4b, which has the line segments concentrated upon two boundary lines. The ϕ histogram is normally distributed while the ρ histogram contains few peaks one for each boundary line. The last two columns show the histograms of the next nested clusters which are normally distributed with less Gaussian noise in the last cluster. This verifies that the Gaussian model is effective in representing the expected noise in our generative model. It also shows that hierarchical clustering is a useful tool for analyzing the collected line segments at different spatial scales.

Given the generative model, we assume that each candidate cluster c has an underlying boundary line that belongs to a normally distributed population with true mean π^*. We only observe random samples from this population where the estimated cluster mean $\widetilde{\pi}$ is an unbiased estimation of π^*. So, we want to verify that $\widetilde{\pi}$ is a reasonable estimate for π^*. In other words, we are interested in measuring the quality that a cluster lies upon a real boundary line which can be written as

$$quality\left(c\right) = P\left(\|\pi^{*} - \widetilde{\pi}\| \leq \epsilon\right) \tag{8}$$

$$= \prod_{\widetilde{x}\in\widetilde{\pi}} \int_{\widetilde{x}-\epsilon}^{\widetilde{x}+\epsilon} \mathcal{N}\left(u|\mu_{\widetilde{x}}, \sigma^2_{\widetilde{x}}\right) du. \tag{9}$$

The cluster quality $quality\left(c\right)$ has its highest probability value when $\widetilde{\pi}$ coincides with π^{*}, and the value becomes lower when $\widetilde{\pi}$ moves further away from π^{*}. We also treat $\widetilde{\pi}$ as a vector of independent components, where each component $\widetilde{x} \in \widetilde{\pi}$ has a true Gaussian model with parameters $(\mu_{\widetilde{x}}, \sigma^2_{\widetilde{x}})$, and $\mu_{\widetilde{x}} \in \pi^{*}$. For clarity, we will refer to $(\mu_{\widetilde{x}}, \sigma^2_{\widetilde{x}})$ as (μ, σ^2) in next discussions. Notice that, μ and σ^2 are random variables so $quality\left(c\right)$ is also a random variable, and we can calculate its expectation as

$$\begin{aligned} E\left(P\left(\|\mu - \widetilde{x}\| \leq \epsilon\right)\right) &= E\left(\int_{\widetilde{x}-\epsilon}^{\widetilde{x}+\epsilon} \mathcal{N}\left(u|\mu, \sigma^2\right) du\right) \\ &= \int_{-\infty}^{\infty} d\mu \int_{0}^{+\infty} d\sigma^2 \int_{\widetilde{x}-\epsilon}^{\widetilde{x}+\epsilon} du \mathcal{N}\left(u|\mu, \sigma^2\right) P\left(\mu, \sigma^2|c\right), \end{aligned} \tag{10}$$

and using conditional probability, we have

$$P\left(\mu, \sigma^2|c\right) = P\left(\sigma^2|\mu, c\right) P\left(\mu|c\right) = P\left(\sigma^2|c\right) P\left(\mu|c\right), \tag{11}$$

given that μ and σ^2 are independent variables. So, the integral of equation 10 becomes

$$\int_{-\infty}^{\infty} d\mu \int_{0}^{+\infty} d\sigma^2 \int_{\widetilde{x}-\epsilon}^{\widetilde{x}+\epsilon} du \mathcal{N}\left(u|\mu, \sigma^2\right) P\left(\sigma^2|c\right) P\left(\mu|c\right). \tag{12}$$

We can then evaluate the inner integral as

$$h(\sigma^2, \widetilde{x} - \mu) = \int_{\widetilde{x}-\epsilon}^{\widetilde{x}+\epsilon} du \mathcal{N}\left(u|\mu, \sigma^2\right) \tag{13}$$

$$= \frac{1}{\sqrt{4}}\left(erf\left(\frac{\widetilde{x} - \mu + \epsilon}{\sqrt{2}\sigma}\right) - erf\left(\frac{\widetilde{x} - \mu - \epsilon}{\sqrt{2}\sigma}\right)\right), \tag{14}$$

where erf is the error function. Now, we can write $g(\sigma^2) = P\left(\sigma^2|c\right)$, $f(\mu) = P\left(\mu|c\right)$, and rewrite the integral over σ in Equation 12 as the inner product

$$\left(g.h\right)\left(\widetilde{x} - \mu\right) = \int_{0}^{+\infty} d\sigma^2 g(\sigma^2) h(\sigma^2, \widetilde{x} - \mu), \tag{15}$$

and Equation 10 becomes

$$E\left(P\left(\|\mu-\widetilde{x}\| \leq \epsilon\right)\right) = \int\limits_{-\infty}^{\infty} d\mu\; f(\mu)\; (g.h)(\widetilde{x}-\mu) \tag{16}$$

$$= f *_{\mu} (g.h), \tag{17}$$

where $*_{\mu}$ denotes the convolution operator over μ. This provides a simple expression to evaluate the expected quality of every candidate cluster. Intuitively, a cluster with a small variance and large number of segments should have a higher chance for the estimated mean $\widetilde{x}$ to be closer to the true mean μ, which drives Equation 17 to give high expectation values.

Given that each cluster has random samples drawn from a true population with unknown mean μ and variance σ^2, we can use Bayesian inference to define the posterior probability distribution of σ^2 as

$$P(\sigma^2|c) \propto P(\sigma^2)P(c|\sigma^2) \tag{18}$$

$$\propto \frac{1}{\sigma^{n+2}} e^{-\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2\sigma^2}}, \tag{19}$$

where we used Jeffrey's prior to set $p(\sigma^2) = \frac{1}{\sigma^2}$. In our case, this distribution has also an unknown mean μ and

$$P(\sigma^2,\mu|c) \propto \frac{1}{\sigma^{n+2}} e^{-\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2\sigma^2}} \tag{20}$$

$$\propto \frac{1}{\sigma^{n+2}} e^{-\frac{\sum_{i=1}^{n}(x_i-\widetilde{x})^2}{2\sigma^2}} e^{-\frac{\sum_{i=1}^{n}(\mu-\widetilde{x})^2}{2\sigma^2}}. \tag{21}$$

Now we can calculate the marginal probability distribution for σ^2 as

$$P(\sigma^2|c) \propto \frac{1}{\sigma^{n+2}} e^{-\frac{\sum_{i=1}^{n}(x_i-\widetilde{x})^2}{2\sigma^2}} \int\limits_{-\infty}^{\infty} e^{-\frac{\sum_{i=1}^{n}(\mu-\widetilde{x})^2}{2\sigma^2}} d\mu \tag{22}$$

$$= \frac{1}{\sigma^{n+2}} e^{-\frac{\sum_{i=1}^{n}(x_i-\widetilde{x})^2}{2\sigma^2}} \sqrt{\frac{2\pi\sigma^2}{n}} \tag{23}$$

$$\propto (\sigma^2)^{-\frac{(v+2)}{2}} e^{-\frac{vs^2}{2\sigma^2}}, \tag{24}$$

where $v = n-1$ is the cluster degree of freedom, and $s^2 = \frac{\sum_{i=1}^{n}(x_i-\widetilde{x})^2}{v}$ is the estimated variance. This is actually a scaled inverse chi-squared distribution [24] $\chi^2_{S.inv}(\sigma^2, v, \tau^2)$ with a scaling parameter $\tau^2 = s^2$. Bayesian inference can also be used to get the marginal probability distribution for μ where

$$P(\mu|c) = \int P(\mu,\sigma^2|c)d\sigma^2 \tag{25}$$

$$= \int P(\mu|\sigma^2,c)P(\sigma^2|c)d\sigma^2, \tag{26}$$

given that

$$P(\mu, \sigma^2|c) \propto P(\mu|\sigma^2, c)P(\sigma^2|c), \tag{27}$$

which also implies

$$\frac{1}{\sigma^{n+2}} e^{-\frac{\sum_{i=1}^{n}(x_i-\widetilde{x})^2}{2\sigma^2}} e^{-\frac{\sum_{i=1}^{n}(\mu-\widetilde{x})^2}{2\sigma^2}}$$
$$\propto P(\mu|\sigma^2, c)(\sigma^2)^{-\frac{(v+2)}{2}} e^{-\frac{vs^2}{2\sigma^2}}, \tag{28}$$

and we get

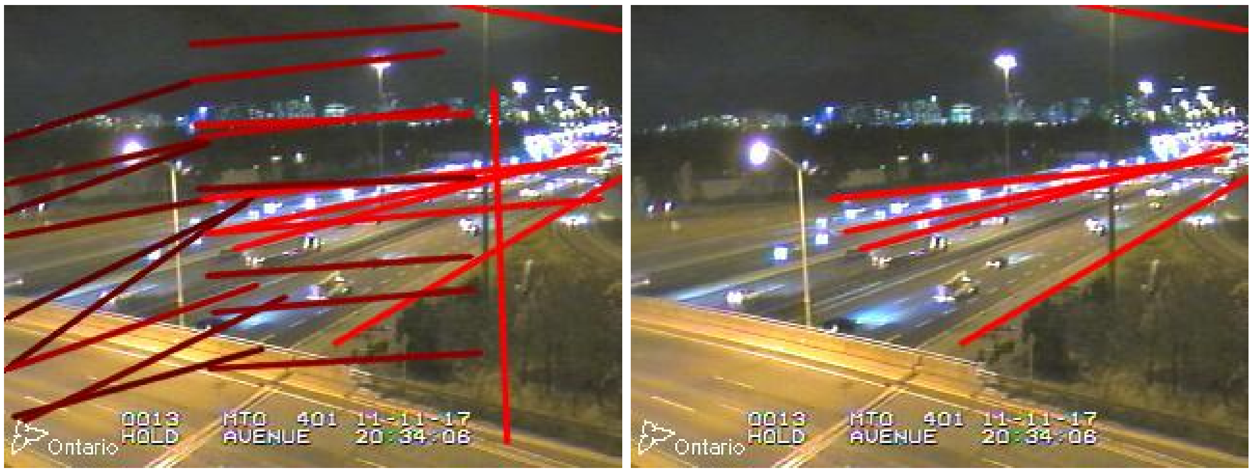
$$P(\mu|\sigma^2, c) = \mathcal{N}\left(\widetilde{x}, \frac{\sigma^2}{n}\right) \tag{29}$$
$$\propto \frac{1}{\sqrt{\sigma^2}} e^{-\frac{n(\mu-\widetilde{x})^2}{2\sigma^2}}. \tag{30}$$

Equation 26, therefore, becomes

$$P(\mu|c) \propto \int_0^\infty \frac{1}{\sqrt{\sigma^2}} e^{-\frac{n(\mu-\widetilde{x})^2}{2\sigma^2}} (\sigma^2)^{-\frac{(v+2)}{2}} e^{-\frac{vs^2}{2\sigma^2}} d\sigma^2 \tag{31}$$
$$\propto \int_0^\infty \sigma^{-v-3} e^{-\frac{1}{2\sigma^2}(n(\mu-\widetilde{x})^2+vs^2)} d\sigma^2 \tag{32}$$
$$\propto \left(1 + \frac{n(\mu-\widetilde{x})^2}{vs^2}\right)^{-(\frac{v+1}{2})}, \tag{33}$$

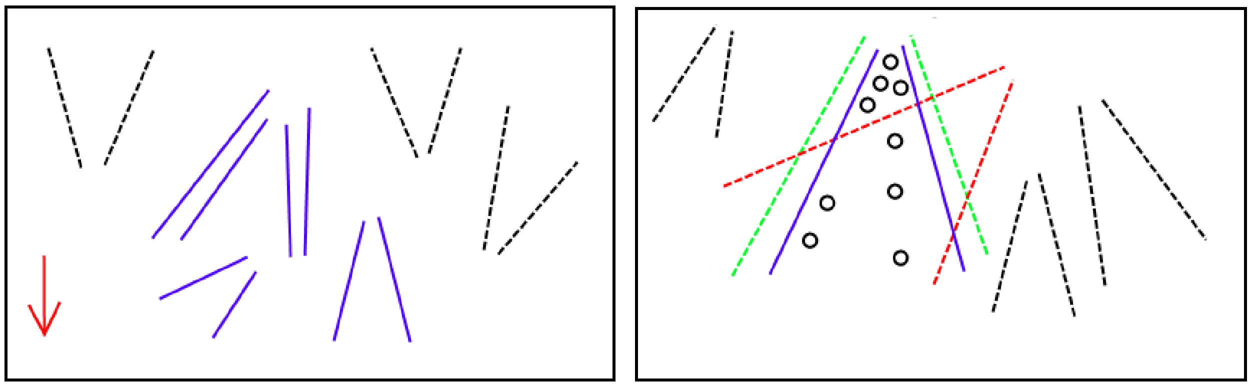
which is simply the standardized Student t distribution [24] where $t = \frac{\mu-\widetilde{x}}{s/\sqrt{n}}$. Now, we can see that the two functions $f(\mu)$ and $g(\sigma^2)$ of Equation 17 represent the marginal distributions of the unknown random variables σ^2 and μ. We also proved that $g(\sigma^2)$ is a scaled inverse chi-squared distribution, and $f(\mu)$ is a student-t distribution. This enables us to define confidence intervals for both σ^2 and μ and numerically evaluates Equation 17. In our experiments, we empirically set the confidence intervals for both μ and σ^2 to 0.95 and $\epsilon = 0.05$.

Equation 17 gives low confidence to 1) clusters with large variance and to 2) clusters with low variance and few samples. At the same time, the equation promotes clusters with low variance and large number of samples. These clusters correspond to the dominant edges accumulated from multiple frames over time. Figure 6 shows the selection of the high confidence clusters over a sample test sequence. Figure 6a shows the mean lines of the top 20% high confidence clusters. These lines are ranked by the confidence value where the more reddish the lines are, the higher the confidence. Figure 6b shows a subset of clusters in Figure 6a that has the largest number of segments. This verifies our assumption of tracking stable edges over time. Notice that, a cluster mean line is generated by projecting the segments on the cluster mean and determining the extent of these projections. This allows the technique to filter out clusters with very small length and generate long road boundaries even if there are missing parts between frames.



(a) (b)

Fig 6: High confidence clusters generated from a 25 frame sequence. (a) Mean lines corresponding to clusters containing at least 20 segments. (b) Mean lines corresponding to clusters containing at least 60 segments. The more reddish the line, the higher the cluster confidence.



(a) (b)

Fig 7: Using perspective filtering and image activity to rank cluster pairs. Solid lines indicate mean lines for cluster pairs that survived the perspective filtering and activity ranking. (a) Perspective filtering where the red arrow indicates the downward direction. (b) Activity based filtering where circles represent vehicular traffic.

3.4 Pairwise Ranking

The detected mean lines of high confidence clusters are usually the dominant lines in an image sequence, but we need to filter these lines to identify the road and lane boundaries. To perform this filtering, a heuristic is applied that constructs cluster pairs and ranks them based on the camera perspective and the image activity cues. The top-ranked pairs are then selected to represent the dominant road and lane boundaries.

Pairwise ranking (Figure 7) starts by computing the vanishing point for each cluster pair. Then, the camera perspective view is used to filter out the pairs which have their vanishing point heading downward (see Figure 7a). The survived pairs are then ranked based on a confidence score r_{Conf}

and an activity score r_{Activity}, where the overall rank for a cluster pair (i, j) is given by

$$r(i,j) = r_{\text{Conf}}^{(i,j)} \times r_{\text{Activity}}^{(i,j)}, \tag{34}$$

where

$$r_{\text{Conf}}^{(i,j)} = \text{Conf}(c_i) \times n_i \times \text{Conf}(c_j) \times n_j, \tag{35}$$

and

$$r_{\text{Activity}}^{(i,j)} = \frac{\text{\#objects within } (i,j) \text{ region}}{\text{area spanned by } (i,j) \text{ region}}, \tag{36}$$

where $\text{Conf}(c_i)$ defines the confidence assigned by equation 8 to cluster c_i. The area spanned by a cluster pair (i, j) is the area of the image region enclosed by the mean lines of the cluster pair (c_i, c_j). The confidence score $r_{\text{Conf}}^{(i,j)}$ encourages pairs with a large number of segments; whereas the activity score prefers pairs that enclose higher activity. We measure the image activity[†] by first applying background subtraction to detect the moving objects [25], then counting the number of objects inside each cluster pair.

4 Experimental Results

We evaluate the proposed method on two datasets recorded from 15 cameras mounted along Ontario's 401 highway. The first dataset consists of video sequences consisting of 50 frames each from 14 camera locations. Each sequence contains 25 frames recorded during the day and an equal number recorded during the night. This dataset is captured at 320×240 resolution. The second dataset consists of one 1627 frames long video that shows camera viewing direction changes and transition from daytime to nighttime. This dataset is recorded at 704×480 resolution. Together the datasets exhibit a wide range of environmental and lighting conditions observed in traffic cameras, including harsh shadows, unlit roads that are only visible in headlights of passing vehicles, headlight glare, camera shake, variations in traffic density, etc. The second dataset also shows the effects of switching camera viewing directions mid-operation. For both datasets, the frames are captured 15 to 20 minutes apart. The frames are manually annotated to generate ground truth. For the results presented here, we set the number of superpixels equal to 100. The online hierarchical clustering parameters are chosen as follows: $\beta = 2$, $\lambda = 0.2$, and $\Delta t_c = 3$ (seconds). We set the clustering tree height to 9.

We compare our method against 1) the Gabor filter based technique (Gabor) that appeared in [7] and 2) the deep learning based scheme (CN24) [10]. CN24 requires a pre-trained network. We use the network learnt in [10] for our purposes. Specifically, CN24 computes a confidence map that assigns to each pixel the likelihood of belonging to the road region. We normalize the confidence map returned by CN24 and consider all pixels with likelihood values equal to or greater than 0.5 to be belonging to the road region. Experiments are carried out on a 2.9 GHz quad-core AMD Athlon processor. Our system is implemented in C++ and Go. It uses multi-threading. The algorithm runs as a pipeline of non-blocking concurrent routines, implementing individual processing steps of the algorithm (super-pixel generation, clustering, ranking, etc.). For Gabor and CN24, we use the code provided by respective authors. Previously, in [11], we proposed three strategies for ranking individual and pair-wise clusters (i.e., boundaries): 1) using χ^2 and Student t test to rank individual

[†]The assumption is that dominant image activity is the result of vehicular traffic in traffic surveillance images.

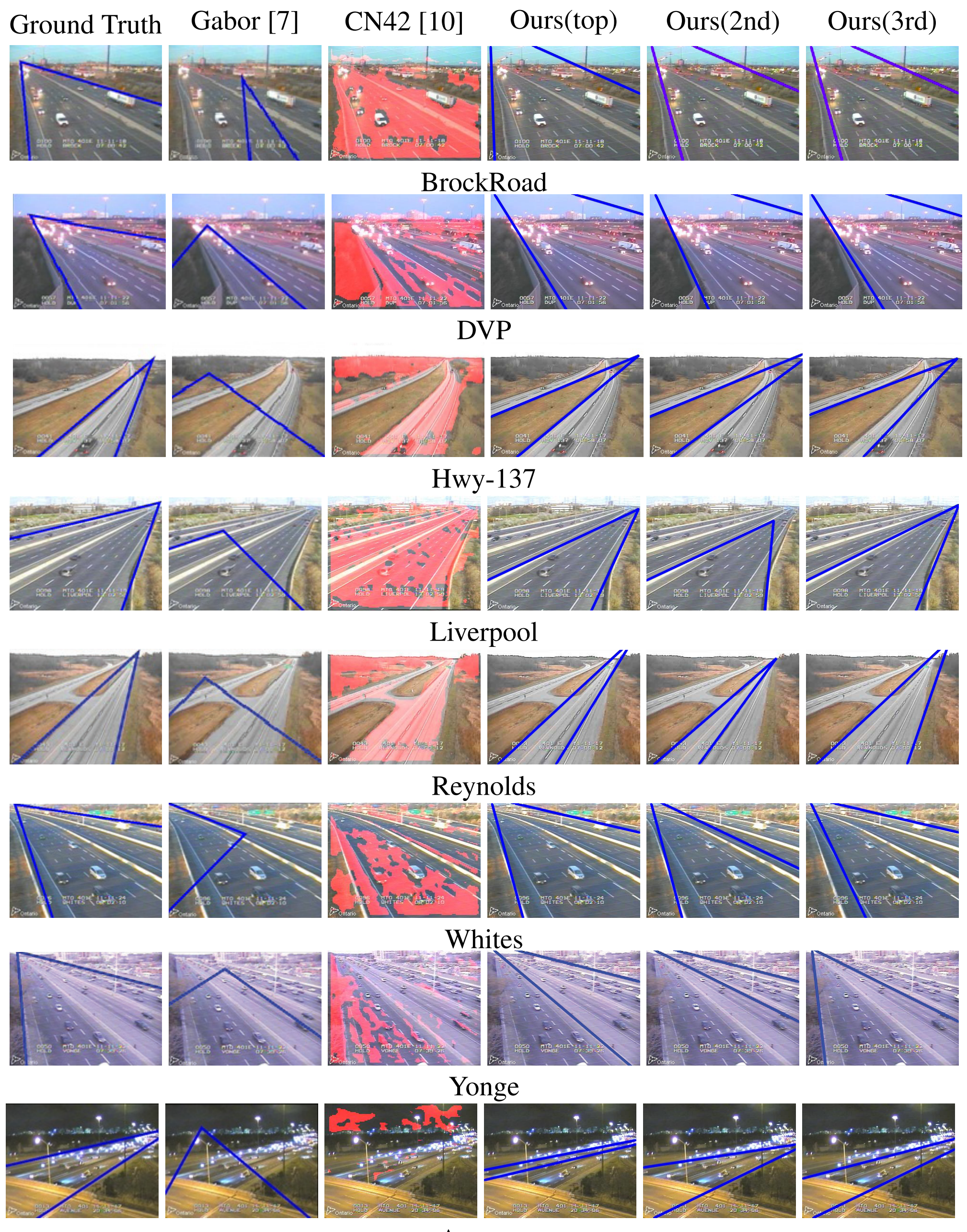


Fig 8: Dataset 1 daytime. Results of the proposed technique in eight different camera locations, compared to the Gabor-based method [7], and the deep learning CN24 method [10]. The first seven locations show day lighting scenarios, whereas the last location presents an occluded road boundary scenario. The first column is the ground truth; the second and the third are the results of [7] and [10], respectively. Notice that the classified road regions of [10] are highlighted by red. The fourth column is our top-ranked pair, and the remaining columns are our top pairs upto the 3rd rank.

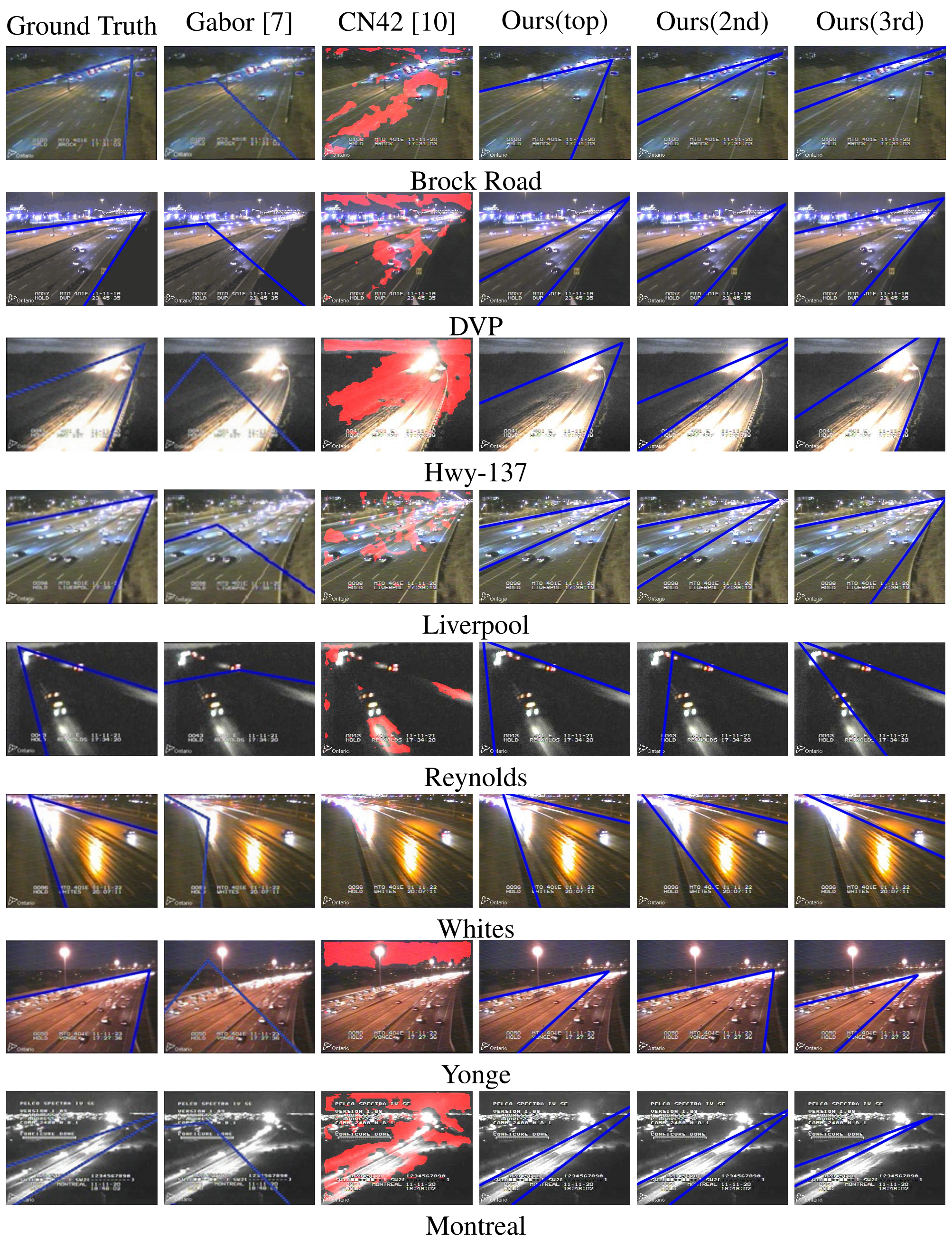
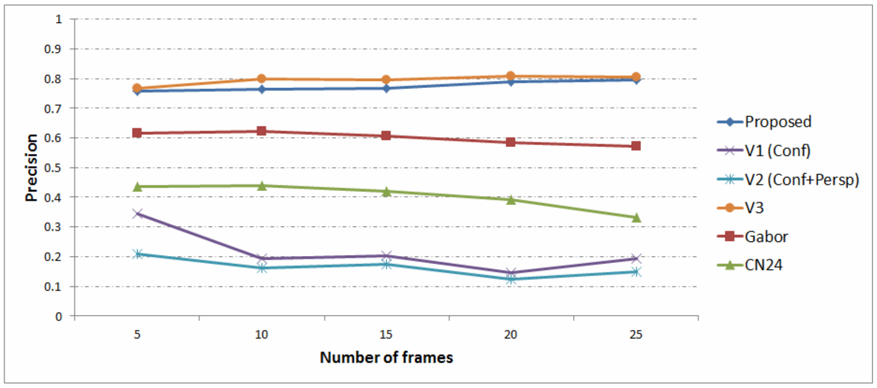
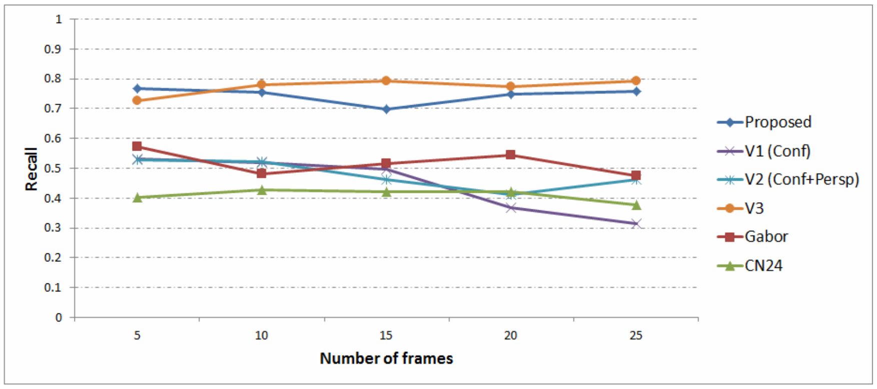


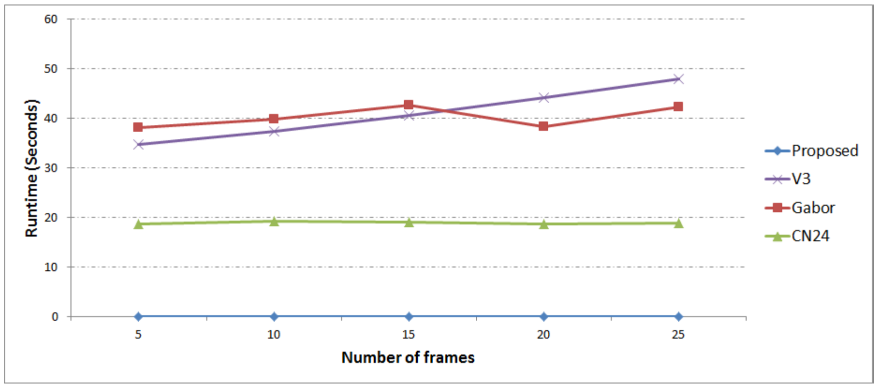
Fig 9: Dataset 1 nighttime. Results of the proposed technique in eight different camera locations during night light, compared to the Gabor-based method [7], and the deep learning CN24 method [10]. The first column is the ground truth; the second and the third are the results of [7] and [10], respectively. Notice that the classified road regions of [10] are highlighted by red. The fourth column is our top-ranked pair, and the remaining columns are our top pairs upto the 3rd rank.



(a) Precision



(b) Recall



(c) Runtime

Fig 10: Dataset 1. Precision, recall and runtime comparisons between different versions of our method, the Gabor filter based method [7], and the convolutional network classification method (CN24) of [10]. The V1 (Conf) method represents our results based on only cluster confidence ranking, whereas the V2 (Conf+Persp) shows our results after both perspective filtering and cluster confidence ranking. The V3 method represents our previous results in [11] using bottom up hierarchical clustering. The proposed method combines V2 with activity ranking. While the proposed method achieves accuracies comparable to those for V3, the proposed method is significantly faster than V3.

clusters; 2) using (1) and perspective cues to rank pair-wise clusters; and 3) using (1), (2), and vehicle activity to rank pair-wise clusters. To make this section more accessible, we identify the three versions of our previous method (which uses bottom-up hierarchical clustering) as V1, V2, and V3. Here, V1 uses only (1) for ranking clusters, V2 uses both (1) and (2) for ranking pair-wise clusters, and V3 uses (1), (2), and (3) for ranking clusters. Recall that the method proposed here uses online hierarchical clustering algorithm. We expect the performance of the method proposed here to be different from V3.

4.1 Dataset 1

Figure 8 shows the detected road regions in 8 (out of 14) daytime sequences. Note also that the Avenue location (last row) contains a partially occluded highway. The first column represents manually labeled ground truth. The next two columns show road regions returned by Gaber [7] and CN24 [10] methods, respectively. The last three columns show results of our method (the three top-ranked road regions computed by our method). Notice how our method is able to pick the road region even when the road is partially occluded.

Figure 9 shows the detected road regions for the 8 (out of 14) locations at nighttime. Again, the left column contains ground truth. Gabor [7] results are presented in column 2 and CN24 results are in column 3 [10]. The last three columns show the three top-ranked pairs returned by our method. This clearly is a challenging sequence. In some cases road is barely visible. Consider Reynolds location (row 5) and Montreal location (row 8), for example, where the road is only visible in vehicle headlights. Despite challenging environmental and lighting conditions, our method does a much better job of identifying road regions than competing Gabor and CN24 methods.

Using the manually labeled data as ground truth, we also compute precision, recall [26], and runtime numbers for ours, Gabor, and CN24 methods. The precision is defined as the ratio of the intersection area between the estimated boundary and ground truth to the area of the estimated boundary. The recall is the ratio of the overlapped area between the detected road boundary and the ground truth to the area of the ground truth. Figure 10 shows the precision, recall, and run-time comparisons, which indicate that the proposed method (when using both perspective and activity cues for ranking pair-wise clusters) outperforms the Gabor and CN24 schemes. Our method also outperforms V1 and V2 by a large margin in terms of precision and recall. Still the precision and recall numbers for the proposed method are similar to V3. Notice, however, that the proposed method is order of magnitude faster than Gabor, CN24, and V3. We do not plot runtimes for V1 and V2, since their accuracy (as seen from precision and recall) is poor. The proposed method uses the same individual and pair-wise ranking cues as V3, so the numbers for V1, V2, and V3 serve to indicate the role and importance of cluster ranking and perspective and activity cues for our proposed method. For these figures the number of frames seen in horizontal axes refer to the size of the sliding window over which evidence is accumulated.

Table 1 summarizes the mean and standard deviation statistics for precision and recall numbers. Our method achieves an average of 77% and 75% for precision and recall, respectively. The Gabor method, on the other hand, achieves 60% and 52% for precision and recall, respectively. Precision and recall numbers for CN24 are 40% and 41%, respectively. Precision and recall numbers for V1 and V2 are low, suggesting that activity information gleaned from the image is an important cue to rank road region candidates. V3 achieves the best precision and recall numbers at 80% and 77%. The reason for this is that the proposed method, which uses an online hierarchical clustering

Methods	Precision	Recall	Average Runtime (seconds)
Proposed	$77\%_{\pm 24\%}$	$75\%_{\pm 25\%}$	0.05
V1 (Conf)	$24\%_{\pm 36\%}$	$45\%_{\pm 39\%}$	0.05
V2 (Conf+Persp)	$16\%_{\pm 25\%}$	$48\%_{\pm 37\%}$	0.05
V3 [11]	$80\%_{\pm 25\%}$	$77\%_{\pm 37\%}$	40
Gabor [7]	$60\%_{\pm 31\%}$	$52\%_{\pm 31\%}$	40.2
CN24 [10]	$40\%_{\pm 21\%}$	$41\%_{\pm 34\%}$	18.8

Table 1: Dataset 1. A summary of the mean and standard deviation statistics for the precision and recall comparisons shown in figure 10. Moreover, we show the average runtime (in seconds) of each compared method.

algorithm, is only able to construct approximate cluster hierarchy as opposed to V3, which uses bottom-up agglomerative clustering to create the *true* cluster hierarchy. The proposed method, however, is an order of magnitude faster than V3. The current streaming implementation of the proposed method achieves speedups of upto 300 times as compared to CN24 and upto 800 times as compared to Gabor and V3.

4.2 Dataset 2

This dataset is recorded by a camera over the course of three days and it shows 8 instances of changing the viewing direction of this camera. Changes in viewing directions are either initiated by human operators or occur at predefined times. Such viewing direction changes do not pose a challenge for Gabor and CN24 methods, since these schemes process each frames individually. Changes in viewing directions, however, pose a problem for V3 and the proposed method, which accumulate evidence over multiple frames. V3 deals with this situation in an ad hoc manner by estimating the road region (from scratch) after every 25 frames. The proposed method, however, is able to maintain road regions through viewing direction changes by "forgetting" stale evidence.

Figure 11 shows the results of our method on example frames selected from the sequence; notice the changes in viewing directions. The columns (one to five) show results for ground truth, Gabor, CN24, V3, and the proposed method, respectively. Rows indicate different viewing directions. Gabor performs poorly; however, CN24 gives decent results, but fails on nighttime frames. Looking at the last two columns the proposed method, which does a better job of dealing with camera viewing direction changes, outperforms V3. Table 2 also confirms this observation. The average precision and recall numbers of the proposed method are 90% and 71%, respectively. These numbers for Gabor are 78% and 43%, respectively. Average precision for CN24 is 64%, and the average recall for CN24 is 49%. For V3, the numbers are 47% and 76%, respectively. Clearly, the proposed method is able to deal with mid-operation camera movement. The proposed method not only has better accuracy, it also has significantly better runtimes. The average (per frame) runtime for the proposed method is 0.13 seconds, which is roughly ≈ 1500 times faster than that of V3, ≈ 400 times faster than that of Gabor, and ≈ 600 times faster than that of CN24.

5 Discussion

We evaluate our method on two real-world datasets. BrockRd, Yonge, and LiverPool cameras show different lighting conditions. Hwy-137 location is unlit and the road is only visible in vehicle headlights. Locations LiverPool, Yonge, and Whites show strong headlight reflections in

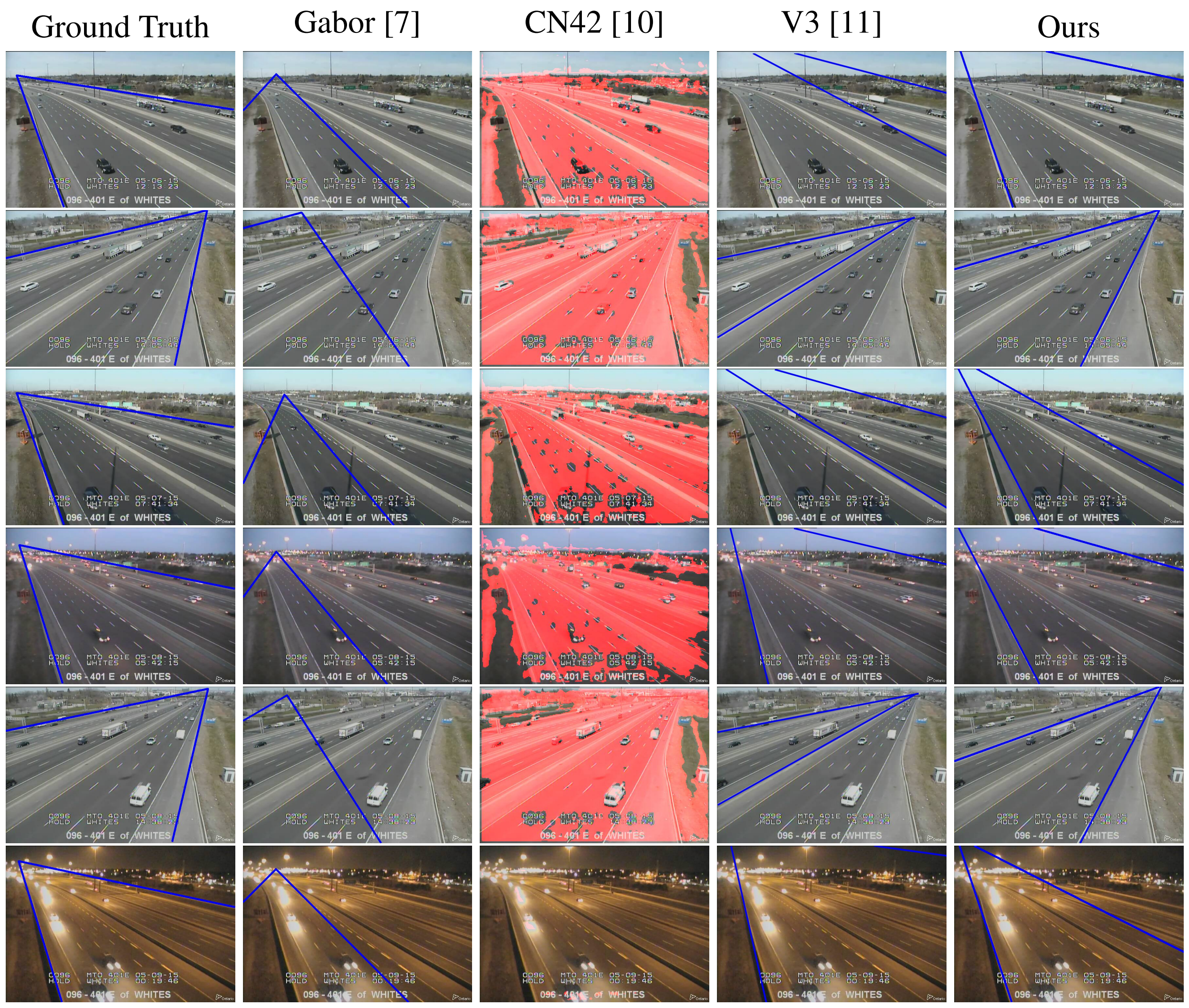


Fig 11: Dataset 2. Results of the proposed technique in six different camera viewing directions of the long video stream dataset, compared to the Gabor-based method [7], the deep learning CN24 method [10], and the V3 algorithm [11]. The first column is the ground truth and the next three columns show the results of [7], [10] and [11], respectively. Notice that the classified road regions of [10] are highlighted by red. The last column is our top-ranked pair.

wet conditions. Avenue location shows a partially occluded highway. The results suggest that our method is able to identify dominant road boundaries under challenging environmental and lighting conditions.

Results listed in Table 1 showcase the importance of activity cue for dominant road region detection. Methods V1 and V2, which ignore activity cue for pair-wise ranking, performs worse than V3 and the proposed approach. V1 favors stable "boundaries," which may or may not be a road boundary. V2 adds perspective filtering to V1; however, this does not result in better performance. V3, which uses activity cue, does well in detecting road regions. The proposed method that uses the same strategy for ranking individual and pair-wise clusters as that of V3 also performs well in detecting road regions. The proposed method also outperforms Gabor and CN24.

The proposed method is significantly faster than V3, Gabor and CN24. It also uses online

Methods	Precision	Recall	Average Runtime (seconds)
Proposed	$90\%_{\pm 11\%}$	$71\%_{\pm 20\%}$	0.13
V3 [11]	$47\%_{\pm 33\%}$	$76\%_{\pm 22\%}$	200
Gabor [7]	$78\%_{\pm 14\%}$	$43\%_{\pm 12\%}$	54.9
CN24 [10]	$64\%_{\pm 26\%}$	$49\%_{\pm 43\%}$	81.5

Table 2: Dataset 2. Mean and standard deviation statistics for the precision and recall comparisons on Dataset 2. The last columns shows average runtime in seconds for each method.

(top-down, approximate) hierarchical clustering, which is much more efficient than computing bottom-up hierarchical clustering. The proposed approach also addresses a limitation of V3. V3 cannot remove or update information within the cluster hierarchy. Consequently, V3 is unable to deal with cameras whose viewing direction may change. Also, V3 performs poorly on stream segments that have few scene activity as it independently estimates road region for every 25 frames. Moreover, the computational time of V3 grows rapidly for higher resolution images because the clustering hierarchy accumulates a larger set of edges. These limitations results in the degraded precision of results and the increased computational cost on Dataset 2. The online hierarchical clustering algorithm employed in the proposed method is able add, update, or remove information in the cluster hierarchy. Precision and recall numbers for the proposed method are similar to those for V3 on Dataset 1, while outperforming V3 on Dataset 2.

Gabor method did not do well on dataset 1. Dataset 1 contains low-resolution frames, which creates a problem for Gabor method. Remembering that this method uses the output of Gabor filtering to define a vanishing point and subsequently exploits this information to estimate the road boundary. In case of possibly noisy, low-resolution images, the vanishing point estimation is wrong, which leads to poor performance on road region detection. For dataset 1, CN24 performs poorly on some frames. CN24 network is learnt using road patches. In case of dataset 1, these road patches have low-resolution and may be noisy. This leads to CN24 poor performance for some frames in dataset 1. Dataset 2 is higher resolution and CN24 begins to do well. Still the proposed method outperforms Gabor, CN24, and V3.

6 Conclusion and Future Work

The paper develops a new online method for detecting dominant road regions in traffic cameras. The proposed method is able to process live video streams at "real-time" frame-rates. It is also able to maintain road regions as cameras are re-positioned or their viewing directions changed. We have evaluated our method on two real-world datasets consisting of images captured by traffic cameras mounted along Ontario's 401 Highway. These datasets adequately capture the extreme environmental and lighting conditions seen in traffic cameras, such as harsh shadows, unlit roads, headlight glare, etc. The first dataset contains low-resolution (320×204) images; whereas, the second dataset comprises of higher-resolution (704×480) images. The results show that our method is robust to environmental and lighting conditions encountered in traffic cameras. Also that the proposed method can deal with low-resolution imagery. The ability to deal with low-resolution imagery becomes exceedingly relevant as bandwidth begins to be the bottleneck, say as we move towards larger networks of traffic cameras.

We have compared our technique with two competing schemes—[7] and [10]—and our method outperforms these both in terms of runtime performance and accuracy. We also compare this method with a previous iteration of our work that appeared in [11] and show that the current method outperforms its previous iteration in terms of runtime performance. Additionally, the current technique can maintain dominant road regions through camera movement.

In the future, we plan to investigate various applications of our method for traffic flow analysis: car counting, excessive speeding, careless driving, accidents, etc.

References

[1] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis, "A survey of video processing techniques for traffic applications," *Image and Vision Computing* **21**, 359–381 (2003).

[2] N. Buch, S. Velastin, and J. Orwell, "A review of computer vision techniques for the analysis of urban traffic," *IEEE Transactions on Intelligent Transportation Systems* **12**, 920–939 (2011).

[3] B. Stewart, I. Reading, M. Thomson, T. Binnie, K. Dickinson, and C. Wan, "Adaptive lane finding in road traffic image analysis," in *Proc. of 7th IEEE International Conference on Road Traffic Monitoring and Control*, 133–136, (Napier Univ. Edinburgh) (1994).

[4] J. Melo, A. Naftel, A. Bernardino, and J. Santos-Victor, "Detection and classification of highway lanes using vehicle motion trajectories," *IEEE Transactions on Intelligent Transportation Systems* **7**, 188–200 (2006).

[5] R. Satzoda and M. Trivedi, "Selective salient feature based lane analysis," in *Proc. of IEEE ITSC*, 1906–1911 (2013).

[6] M. Aly, "Real time detection of lane markers in urban streets," in *Proc. of IEEE Intelligent Vehicles Symposium*, 7–12, (Eindhoven, The Netherlands) (2008).

[7] H. Kong, J. Audibert, and J. Ponce, "General road detection from a single image," *IEEE Transactions on Image Processing* **19**, 2211–2220 (2010).

[8] S. Zhou, J. Xi, J. Gong, G. Xiong, and H. Chen, "A novel lane detection based on geometrical model and gabor filter," in *Proc. of Intelligent Vehicles Symposium*, 59–64, (San Diego, CA) (2010).

[9] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using b-snake," *Image and Vision Computing* **22**, 269–280 (2004).

[10] C. Brust, S. Sickert, M. Simon, E. Rodner, and J. Denzler, "Convolutional patch networks with spatial prior for road detection and urban scene understanding," in *Proc. of 10th International Conference on Computer Vision Theory and Applications*, 11–14 (2015).

[11] M. Helala, K. Pu, and F. Qureshi, "Road boundary detection in challenging scenarios," in *Proc. 9th IEEE Conference on Advanced Video and Signal-Based Surveillance*, IEEE Computer Society, (Beijing, China) (2012).

[12] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "Self-adaptive anytime stream clustering," in *Proc. of IEEE ICDM*, 249–258 (2009).

[13] M. Helala, K. Pu, and F. Qureshi, “A stream algebra for computer vision pipelines,” in *CVPRW*, 800–807 (2014).

[14] A. Hillel, R. Lerner, D. Levi, and G. Raz, “Recent progress in road and lane detection: a survey,” *Mabchine Vision and Applications* **25**(3), 727–745 (2014).

[15] Z. Chen, Y. Yan, and T. Ellis, “Lane detection by trajectory clustering in urban environments,” in *Poc. of IEEE ITSC*, 3076–3081 (2014).

[16] J. Alvarez, A. Lopez, T. Gevers, and F. Lumbreras, “Combining priors, appearance, and context for road detection,” *IEEE Transactions on ITS* **15**, 1168–1178 (2014).

[17] S. Paris and F. Durand, “A topological approach to hierarchical segmentation using mean shift,” in *Proc. of IEEE CVPR*, 1–8, (Minneapolis, MN) (2007).

[18] M. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, “Entropy rate superpixel segmentation,” in *Proc. of IEEE CVPR*, 2097–2104, (Colorado) (2011).

[19] C. Chefd’Hotel and A. Sebbane, “Random walk and front propagation on watershed adjacency graphs for multilabel image segmentation,” in *Proc. of IEEE ICCV*, 1–7, (Rio de Janeiro) (2007).

[20] S. Yu and J. Shi, “Multiclass spectral clustering,” in *Proc. of IEEE ICCV*, **1**, 313–319, (Nice, France) (2003).

[21] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinsonl, and K. Siddiqi, “Turbopixels: Fast superpixels using geometric flows,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31**, 2290–2297 (2009).

[22] M. Helala, M. Selim, and H. Zayed, “A content based image retrieval approach based on principal regions detection,” *International Journal of Computer Science Issues* **9**, 204–213 (2012).

[23] L. Figueiredo, “Adaptive sampling of parametric curves,” in *Graphics Gems V*, 173–178, Academic Press (1995).

[24] A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin in *Bayesian Data Analysis*, Florida : CRC Press (2004).

[25] K. Kyungnam, T. Chalidabhongse, D. Harwood, and L. Davis, “Background modeling and subtraction by codebook construction,” in *Proc. of Int. Conf. ICIP*, **5**, 3061–3064, (Singapore) (2004).

[26] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, New York: ACM Press, Addison-Wesley (1999).

Mohamed A. Helala is currently a Ph.D. candidate in the department of Computer Science at UOIT. He received a M.Sc. degree in Computer Engineering from Benha University, Egypt in

2010. His research interests include computer vision, machine learning, and data stream processing.

Faisal Z. Qureshi received M.Sc. and Ph.D. degrees in Computer Science from the University of Toronto, Toronto, Canada, in 2000 and 2007, respectively. He is an Associate Professor of Computer Science and the (founding) director of the Visual Computing lab. He joined the UOIT in 2008 from Autodesk Canada Co. in Toronto, where he was a Software Developer. His research interests include sensor networks, computer vision, and computer graphics. He has also published papers in space robotics. He is a member of the IEEE, the ACM and the CIPPRS.

Ken Q. Pu is an Associate Professor of Computer Science at UOIT. He received his PhD in Computer Science from University of Toronto in 2006. His research area is in the theory and applications of database and information systems. His current research interests include data processing in computer vision systems, embedded and pervasive databases, and human data interaction. He is a member of the IEEE and the ACM.

List of Figures

List of Tables