CIVIL ENGINEERING STUDIES
Illinois Center for Transportation Series No. 18-012
UILU-ENG-2018-2012
ISSN: 0197-9191

OPPORTUNISTIC TRAFFIC SENSING USING EXISTING VIDEO SOURCES (PHASE II) FINAL REPORT

Prepared By
Jakob Eriksson
Yanzi Jin
University of Illinois at Chicago

Research Report No. FHWA-ICT-18-010

A report of the findings of

ICT PROJECT R27-169
Opportunistic Traffic Sensing Using Existing Video Sources (Phase II)

Illinois Center for Transportation

August 2018

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. FHWA-ICT-18-010	**2. Government Accession No.** N/A	**3. Recipient's Catalog No.** N/A
4. Title and Subtitle Opportunistic Traffic Sensing Using Existing Video Sources (Phase II) Final Report		**5. Report Date** August 2018
		6. Performing Organization Code N/A
7. Author(s) Jakob Eriksson and Yanzi Jin		**8. Performing Organization Report No.** ICT-18-012 UILU0ENG-2018-2012
9. Performing Organization Name and Address Illinois Center for Transportation Department of Civil and Environmental Engineering University of Illinois at Urbana-Champaign 205 North Mathews Avenue, MC-250 Urbana, IL 61801		**10. Work Unit No.** N/A **11. Contract or Grant No.** R27-169
12. Sponsoring Agency Name and Address Illinois Department of Transportation Bureau of Research 126 East Ash Street Springfield, IL 62704		**13. Type of Report and Period Covered** Final Report February 16, 2017 – August 15, 2018 **14. Sponsoring Agency Code** FHWA

15. Supplementary Notes
Conducted in cooperation with the U.S. Department of Transportation, Federal Highway Administration.

16. Abstract
The objective of this project was to produce accurate vehicle counts and turn counts from opportunistically acquired video sources such as surveillance, red light, and other traffic cameras, using computer vision techniques. The project scope included algorithm design, software development, and a web portal for practitioner use.

17. Key Words vehicle count, turn count, video, computer vision	**18. Distribution Statement** No restrictions. This document is available through the National Technical Information Service, Springfield, VA 22161.

19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 18 pp	**22. Price** N/A

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized**

ACKNOWLEDGMENT, DISCLAIMER, MANUFACTURERS' NAMES

This publication is based on the results of **ICT-R27-169, Opportunistic Traffic Sensing Using Existing Video Sources**. ICT-R27-169 was conducted in cooperation with the Illinois Center for Transportation; the Illinois Department of Transportation; and the U.S. Department of Transportation, Federal Highway Administration.

Members of the Technical Review panel were the following:

- William Morgan (IDOT)
- Jenni LeSeure, IDOT
- Mike Miller, IDOT
- Vince Durante, IDOT
- Abraham Emmanuel, Chicago DOT

EXECUTIVE SUMMARY

The objective of this project was to investigate new and existing computer vision algorithms for the purpose of producing vehicle counts and turn counts from opportunistically acquired, existing video sources. Included in the scope was also the development of software capable of performing such task, and a web portal providing an interface to this software, to be made available to practitioners.

This report addresses the second phase of the project, which focused on several refinements to the algorithms used, in order to improve accuracy as well as robustness to adverse environmental and lighting conditions. This phase also introduced a web portal interface for using the system remotely.

Analyzing opportunistically collected video presents several interesting challenges to videos recorded for the purpose of traffic analysis. Opportunistic video comes in a variety of formats and qualities. The perspective offered by the video is often not ideal, including substantial occlusion from obstacles, or vantage points located between vehicles due to a low angle of observation.
This type of videos can be collected under any condition including severe changes of seasons, weather, and lighting. Finally, opportunistic video is available in virtually unlimited quantity, which means that processing needs to be efficient in order to take advantage of this resource.

The resulting video analysis system operates in three main phases. First is the scene-learning phase, which analyzes the scene of a camera from a ``topic'' perspective. Here, individual short video sequences (snippets) are compared against other snippets, using a technique called topic modeling. This technique was adapted for use in computer vision from the field of text mining. Topic modeling produces a set of movement patterns that frequently occur in a given scene, such as “driving east”, or “turning right onto the northbound street.” The output from this phase is a set of such movements, with information about vehicle size, centerline, and entry/exit points. Ideally, scene learning is applied to a video captured during typical benign conditions.

Tracking, which is the second phase, identifies and follows (tracks) individual cars as they traverse the scene. Tracking takes video and scene-learning information as input and produces a set of “trajectories”, which in turn are a series of bounding boxes (one per video frame). A trajectory is used to describe the movement of an observed object throughout a scene. Tracking uses prior knowledge about the scene to decide which objects to track and which objects to recover from temporary loss of tracking.

The final phase, counting, compares the trajectories produced by tracking, against counting templates, which are either hand-drawn by the user, or automatically produced by the scene-learning phase. In both cases, the templates are reviewed and named by the user in order to produce human-legible statistics. The final vehicle counts are produced in spreadsheet format.

TABLE OF CONTENTS

CHAPTER 1: SCENE LEARNING

To improve overall tracking and counting accuracy, the research team introduced a scene-learning component. Scene learning analyzes video independent of object tracking, to learn about typical movements within the scene from long-term statistics. Scene learning provides two advantages. First, with a stationary camera, the scene contains many long-term features that cannot be inferred in the short term; such as the expected size, speed, and movements of vehicles in the scene. Since these do not change significantly over time, scene learning can be performed during benign conditions, and the inferred knowledge can be used under all conditions. Second, in contrast with computing expected/average trajectories based on tracking results, the scene-learning approach is independent of the tracker's performance. Thus, tracking errors do not cause scene-learning errors. The underlying technique used for scene learning is called a topic model.

1.1 TOPIC MODEL

Scene learning is becoming a hot topic due to its wide range of applications, such as traffic analysis, visual query, and anomaly detection. It provides a higher-level summary without knowing the individuals in the scene. There are some well-studied methods that could be used for motion pattern learning, such as trajectory clustering. However, it requires good tracking results, which is not available for most surveillance videos. Alternatively, instead of clustering trajectories, there has been work on clustering optical flow tracklets or optical flow vectors, where the clusters are frequent motions in a video. These works, such as (Wang et al. 2009), often use nonparametric Bayesian methods such as the topic model for clustering and they usually work well on finding the motion topics. This approach was adopted in this project with further post processing for the vehicle tracking/counting framework.

1.1.1 Video Representation

The videos were split the into short clips, each of them was about 90 frames (~ 3 sec). Each frame was split into 10×10 pixel grids. The optical flow between consecutive frames on each grid was computed and then quantized into twelve directions. Unlike previous methods, twelve directions were used in order to cover more than just horizontal and vertical movements. Although this can result in a much larger vocabulary, the learned motions make more sense for the future applications. Make an analogy with to document clustering, a video is a corpus containing many documents (video clips) and the quantized optical flow vectors correspond to words in the documents. For example, for a video of size 320×240 in pixel, there will be a vocabulary dictionary size of $32 \times 24 \times 12$. Counting the frequency of each vector in a video clip provides a bag-of-words feature for the document. Note that the bag-of-words representation treats each word independently.

1.1.2 Hierarchical Dirichlet Process (HDP)

In most clustering methods, the number of clusters should be provided, and the results may vary according to the number of clusters. Since this is inaccurate for complex videos and requires human input, the Hierarchical Dirichlet Process (HDP) was used, which is a non-parametric clustering method that required no specification of cluster numbers, commonly used for document clustering. It is a

generalization of Dirichlet Process (DP). In this model, the visual words are groups of observations, each exhibiting a mixed proportion of shared mixture components. The mixture components are learned in this model, also called "topics".

An HDP is a two-level DP with shared parameters. Each group j is associated with a draw from a shared DP whose base distribution is also a draw from the top-level DP.

$$G_0 \sim DP(\gamma, H)$$

$$G_j \mid G_0 \sim DP(\alpha_0, G_0), \text{ for each } j,$$

Here j is the group index. At the top-level, the distribution G_0 is drawn from a DP with concentration parameter γ and base distribution H. The base distribution H is a symmetric Dirichlet over the vocabulary simplex. Since the numbers drawn from a Dirichlet distribution sum up to 1, they are usually used as the parameter of a multinomial distribution. Each atom is a distribution over the vocabulary, the atoms $\boldsymbol{\phi} = (\phi_k)_{k=1}^{\infty}$ are drawn independently $\phi_k \sim Dirichlet(\eta)$. Simply speaking, G_0 is a discrete distribution over the atoms $\boldsymbol{\phi}$. At the bottom level, G_0 is used as the base distribution to draw each group distribution G_j. By such hierarchical definition, the atoms are shared among G_j, which makes sense that different documents may belong to the same topic.

In this setting, a group is a visual document and the ith word is drawn from jth document x_{ij} as follows,

$$\theta_{ji} \sim G_j, \qquad x_{ij} \sim Multi(\theta_{ji}).$$

Here θ_{ji} is the topic assignment of x_{ij}, which is associated with a multinomial distribution $\phi_{\theta_{ji}}$. One could potentially have an infinite number of atoms. However, only a finite number of atoms/topics will be learned. After Gibbs sampling, there is a multinomial distribution for each topic over the vocabulary where each word in a document has a topic assignment. Figure 1 visualizes those topics learned by HDP. The color indicates the direction shown in the color wheel on the right and the lightness indicates the value of the multinomial at the corresponding cell. The Figures show that HDP does a good job in summarizing the motions in the video, even without knowing any moving objects in the scene. For more details on Gibbs sampling, please see (Teh et al. 2012).

Figure 1: Motion topics learned by HDP, the color indicates direction, lightness indicates multinomial parameter value for the word entry.

1.2 CENTERLINE EXTRACTION

1.2.1 Ridge Extraction

The ridge climbing method was adopted for this section (Zhao and Wang 2013). Just by looking at the topics in Figure 1, one could roughly infer how the vehicles move in the video. The idea of this method is to start from a high-density point, follow the shape and direction indicated by the visual word, and get a complete ridge of the motion. Due to the different views of the videos, some topics may expand to a wide area, which does not always get a perfect centerline right in the center of the high-density area. Instead, this method was improved by getting multiple ridge lines and then shrink them into a more confident centerline.

Let ϕ be a multinomial distribution of a topic learned by HDP. Each grid i on the frame has values corresponding to D directions $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_D\}$, written as $(\phi_i^{\theta_1}, \phi_i^{\theta_2}, \ldots, \phi_i^{\theta_D})$. In this particular case, $D = 12$. Since the pixels are not likely to move on the opposite directions in a short clip, a reduced distribution on $D/2$ directions with the sign to indicate direction, was considered. For each direction d,

$$\varphi_i^{\theta_d} = \begin{cases} \phi_i^{\theta_d} & \phi_i^{\theta_d} \geq \phi_i^{\theta_{-d}} \\ -\phi_i^{\theta_{-d}} & \phi_i^{\theta_d} < \phi_i^{\theta_{-d}}, \end{cases}$$

where $\phi_i^{\theta_{-d}}$ indicates the value of ϕ in grid i on the opposite direction of θ_d. For the simplicity of notation, we define $\varphi_i = \sum_{d=1}^{D/2} \left| \phi_i^{\theta_d} \right|$.

The ridge climbing process was started from a cell i at (x, y), where we follow the forward direction and go to the next point (x', y'), the procedure repeats as described in the following equation until it converges.

$$\Delta_{\theta_d} = v_{\theta_d} + a \cdot m_{\theta_d}$$

$$x' = x + \sum_{d=1}^{\frac{D}{2}} \Delta_{\theta_d} \cdot \cos\theta_d$$

$$y' = y + \sum_{d=1}^{\frac{D}{2}} \Delta_{\theta_d} \cdot \sin\theta_d$$

To compute Δ_{θ_d}, a small neighborhood was chosen with a size of n around (x, y) and the following values were computed:

$$v_{\theta_d} = \frac{\sum_{i=1}^{n} \varphi_i^{\theta_d}}{\sum_{i=1}^{n} \varphi_i}, \qquad m_{\theta_d} = \frac{\sum_{i=1}^{n} \varphi_i \cdot (x_i - x)}{\sum_{i=1}^{n} \varphi_i}$$

v_{θ_d} determines the step size along the main direction, m_{θ_d} restricts the next step within the high-density area. To look for the source point, the sign of v_{θ_d} was flipped when computing (x', y').

Instead of starting from the cell with global maximal density, we start with many cells with local maximal density φ in its neighborhood. The left image in Figure 2 shows the ridges extracted by the above procedure, they roughly cover the high-density area. Adjusting the neighborhood size may result in getting ridges with different densities. Using a finer quantization of flow words of 12 directions made it possible to trace the ridges in arbitrary directions.

1.2.2 Ridge Clustering

Next, the Adaptive Multi-Kernel-based Shrinkage (AMKS) method was applied to the extracted ridges, which is originally proposed for trajectory clustering in (Xu et al. 2015). Roughly speaking, it is a clustering method in a certain direction. In this case, it was specific to the travelled direction of the ridges. In every iteration, each ridge was moving toward the centerline. Ideally, after convergence, all the ridges will have moved to the centerline. Then the overlapped line was extracted as the final centerline. Before doing the clustering, the overlapped ridges were filtered to reduce the computation overhead. The middle image in Figure 2 gives an example, where the dot at the right end indicates the sink.

1.3 ENTRY-EXIT AREA

As proven later in Chapter 2, trackers need proper initialization and termination in real-world applications. However, automatic entry/exit point learning is rarely addressed in the current literature. There has been some work using the Mixture of Gaussian (MOG) methods to cluster start and end point of trajectories. However, this method is sensitive to noise and it is hard to get clean trajectories without human pruning. On the other hand, the ridges extracted from the ridge climbing method are more reliable. Continued with the above procedure, the source/sink point was fit into a single Gaussian as the entry/exit area, as illustrated by the green and red ellipses in the right image in Figure 2. Note that the ellipses do not indicate the size of the entry/exit area, but the area of source/sink points of most ridges. The density of extracted ridges gives a nice interpretation of the area and it is unlikely for vehicles to go beyond the topic area. Therefore, it may be used as preliminary evidence for initialization.

Figure 2: Ridges (left), centerline (middle) and entry exit (right) extracted by our algorithm.

CHAPTER 2: OBJECT TRACKING

Vehicle tracking has important applications in traffic engineering. However, current tracking algorithms all require initialization as input, leading to semi-automatic tracking systems. To avoid manual input, these trackers rely on background subtraction and/or object detectors for initialization. Here, the primary challenge is robustness to variations in illumination condition, viewpoint, and video quality. The background subtraction model could fail with illumination change, while detectors are not appropriate for detecting a vehicle in the distance, or in a grainy low-resolution video. Additionally, the low throughput of most trackers prevents widely deployed surveillance applications. As pointed out by the VOT 2016 challenge report, none of the top-ranked trackers run in real-time for even a single object.

A fully automatic algorithm was proposed for vehicle tracking that runs faster than real-time. With a sensor fusion approach, the research team combined background segmentation, object detection, and optical flow into a single, robust vehicle tracking system via Kalman filtering. Initialization uses the same three sources to automatically identify moving objects in the scene. Finally, when an object exits the scene, its movements are analyzed to filter out unlikely object trajectories. To evaluate the proposed algorithm as well as prior work, a hand-annotated dataset was created, which consisted of 11 diverse 5-minute videos collected from existing traffic surveillance cameras. For each frame, the location and extent of each moving object was provided, which enabled accurate, quantitative evaluation.

The proposed algorithm was compared against multiple state-of-the-art trackers, which rely on human input for initialization. On this dataset, there was considerably better performance and substantial accuracy improvement when using our new automatic initialization method than that of the state-of-the-art with manual initialization. Moreover, we demonstrate throughput 4 times faster than real time and over 5 times faster compared to 5 out of 7 several baseline trackers. In the best case, the throughput improvement is up to $47\times$.

2.1 PRELIMINARIES

The problem of vehicle tracking in existing traffic surveillance video presents some unique computer vision challenges, including scale changes, video quality (exposure control, automatic white balance, and compression), weather conditions, illumination changes, variations in perspective, and occlusion. Current work in object detection, tracking, and background subtraction can deal with a subset of these conditions, but so far a generic system has been elusive. Below, we first introduce the underlying methods used in our system, then describe our vehicle tracking framework in detail.

2.1.1 Background Subtraction

Background subtraction generates a binary foreground mask given a sequence of frames. Connected areas in the foreground mask can be treated as moving objects, although this technique can be error-prone. ViBe was used for its balance of speed, robustness, and accuracy (Barnich and Droogenbroeck 2011.) However, other methods could be substituted with acceptable results in many cases.

Figure 3 summarizes four common failure cases of the background subtraction with foreground bounding boxes on the original frame on the left and the foreground on the right. Automatic exposure (Figure 3(a)) is performed by the camera during recording, whereas illumination variation (Figure 3(b)) is due to external light sources, such as the sun and vehicle headlights. Both automatic exposure and illumination variation cause rapid and widespread changes in pixel values, which is something that most background subtraction methods struggle with. Ghosting (Figure 3(c)) usually happens when a foreground object remains stationary for a long time, during which time it is gradually assimilated into the background model. When the object begins to move, what appears from behind the object is inaccurately marked as foreground, until the background model has had time to adjust. Occlusion (Figure 3(d)) creates a single connected foreground area out of two or more moving objects. Sometimes it creates multiple foreground areas for a single moving object, breaking any assumption of a one-to-one mapping between foreground areas and moving objects.

(a) Camera auto-exposure.

(b) Illumination change.

(c) Ghosting. Stopped vehicles have become part of the background model.

(d) Occlusion.

Figure 3: Background subtraction failure cases. Pixel values change for reasons other than motion.

In summary, background subtraction provides the ability to capture small movements without manual setup beforehand. However, it is error-prone and must be compensated by other methods to create a robust vehicle tracking system.

2.1.2 Object Detection

Object detectors work on individual frames by scanning the image for areas that appear similar to offline training samples. Compared to background subtraction, object detection method tends to be more robust to illumination change and occlusion. However, the cost of object detection is significant, as it often involves an exhaustive search throughout the image, both in location and object size. More recently, deep neural networks have emerged as a promising approach to object detection. A state-of-the-art detector called faster-RCNN was used (Ren et al. 2017). Running on a high-end graphics processing unit (GPU), the time required for detection on one image drops from 2

seconds to 198 milliseconds (ms) on the PASCAL 2007 dataset, making the real-time detection in video feasible. However, like other detectors, faster-RCNN still has missing and false detections. A missing rate in excess of 65% and 86% on high- and low- resolution videos is reported on the anootated dataset, respectively. Thus, given the high miss rate, especially on poor quality images, object detection alone will not suffice for a robust vehicle tracking system.

2.1.3 Optical Flow

Optical flow is an estimate of the movement of pixels between two images: in our case, two consecutive video frames. Optical flow provides a low-level description of motion in images and can offer useful evidence for tracking applications. Estimating optical flow is a research area in its own right, but the seminal Lucas-Kanade algorithm was used in this system, as it runs fast on GPUs. It also provides useful results while making minimal assumptions about the underlying scene and image. Figure 4 illustrates two optical flow problems that may affect tracking accuracy. The left column shows the direction and magnitude of the optical flow vectors, while the right column is the color code visualization of the optical flow results, with the color wheel at the bottom right corner of Figure 4(b) indicating the corresponding direction. Figure 4(a) illustrates the so-called aperture problem, where the center of the truck has no reported optical flow, due to its large and uniformly colored surface. Figure 4(b) illustrates the "turbulent", error-prone flow that occurs where objects traveling in opposite directions meet.

(a) Aperture problem. (b) Occlusion "turbulence".

Figure 4: Common problems in optical flow estimation.

Thus, while *accurate* optical flow estimates offer valuable information about movement in the scene, they are neither complete (due to the aperture problem), nor free of severe estimation errors, near the occlusion boundaries in particular.

2.1.4 Object Entry and Exit

Currently available datasets usually have tracked objects in the center of the first frame. However, initialization can be more challenging when objects enter the scene in a variety of ways such as: approaching from a distance; entering from the image boundary; appearing from behind an occluding object—moving or stationary; and becoming visible due to changes in lighting or background conditions. Termination has similar challenges—vehicles may disappear temporarily behind obstructions or due to changing conditions, they may linger near the edge of the screen, exit the scene while behind a moving vehicle, or disappear slowly into the distance. It is usually natural to terminate tracking when a sequence ends or when objects leave the scene in short sequences. In

practice, however, it is hard to distinguish between exiting and temporal occlusion when the object is not visible in long-time videos.

As automatic initialization and termination are missing in the majority of current tracking literature, there are generally two accepted methods. The first is to heuristically manually label an entry/exit area beforehand under the assumption that objects always enter or exit within a certain area. The second is to rely on fully manual initialization. The first method is too simplistic for general purpose vehicle tracking, and the second is impractical under constrained expense/time budget for large-scale use.

In the only available literature (Y.Wu et al. 2013) that explicitly addresses the effect of tracker initialization, the author concludes that slight temporal and spatial variation would result in performance difference. However, unlike the currently available dataset, vehicles frequently encounter significant scale change while leaving or entering the scene in the traffic surveillance videos are available. This presents a unique problem for initialization because early initialization results in small poor-quality images, and late initialization results in missing information due to the short trajectory. Thus, striking the right balance between tracking performance and lifetime is the key to automatic tracker initialization.

2.2 PROPOSED METHOD

Figure 5 describes the workflow of the automatic tracking application. First, background subtraction and object detection was applied to each frame. This generated two sets of candidate boxes, which were used to initialize and update trackers. Each tracker is represented by an individual Kalman. Optical flow was also computed. Any flow that matched a tracked object, both by location and velocity, was used for tracker update. Tracking was terminated based on object location, velocity and time; short-lived or otherwise spurious objects are filtered out.

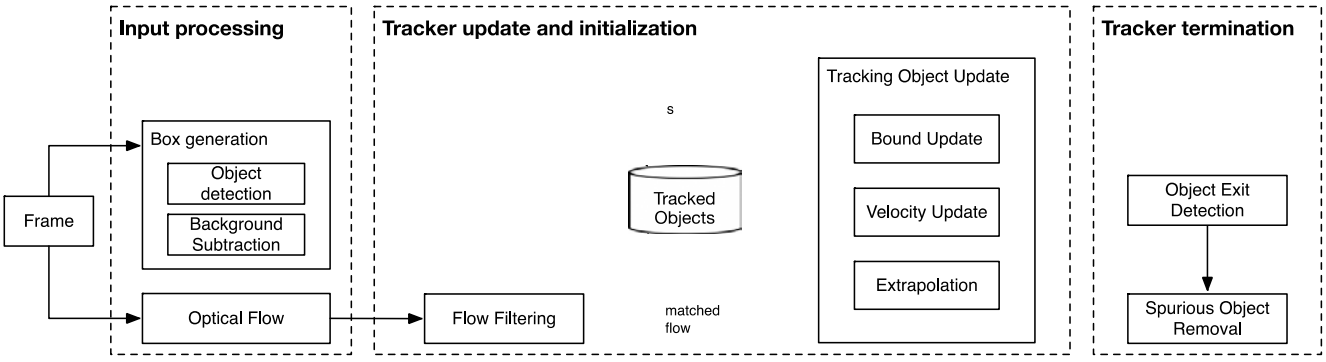

Figure 5: Overview of proposed system. Separate Kalman filter state is initialized, maintained and terminated for each object, updated by background subtraction, object detection and optical flow.

2.2.1 Model Definition

The Kalman filter was used to smooth out the noises in the observed measurements by the aforementioned components. More importantly, the filter was used to integrate the strengths and

make up for the weakness of each component. The prediction by its linear model was corrected with measurements observed over time. Therefore, the generated estimation is much smoother despite the noises in measurement input. For such a continuous system, a time unit was defined as dt, which is the time interval when updates were performed, and in this case, the time between two consecutive frames. Each variable had its own value at a certain time step t, indicated by the subscript. The prediction was performed as follows:

$$\widehat{\boldsymbol{x}}_t^- = \boldsymbol{A}\widehat{\boldsymbol{x}}_{t-1}^-, \qquad \boldsymbol{P}_t^- = \boldsymbol{A}\boldsymbol{P}_{t-1}\boldsymbol{A}^T + \boldsymbol{Q}.$$

Here $\widehat{\boldsymbol{x}}_{t-1}^-$and $\widehat{\boldsymbol{x}}_t^-$ are internal states before and after prediction at time t. $\boldsymbol{P}_{t-1}$ and $\boldsymbol{P}_t^-$ are prior and post error covariances, and $\boldsymbol{Q}$ is the process noise covariance. In our case, we define the internal state a 10-dimensional vector:

$$\boldsymbol{x} = [x, y, w, h, x', y', w', h', x'', y''],$$

corresponding to the object's top-left location (x, y), size (w, h), as well as the velocity (x', y') rate of growth (w', h') and acceleration (x'', y''), respectively. The dynamic model A is defined based on the physics equation of displacement with velocity and acceleration. With the assumption that the object has constant acceleration within dt, we have:

$$x_t = x_{t-1} + x'_{t-1} \cdot dt + \frac{1}{2}x''_{t-1} \cdot dt^2$$

$$y_t = y_{t-1} + y'_{t-1} \cdot dt + \frac{1}{2}y''_{t-1} \cdot dt^2$$

$$x'_t = x'_{t-1} + x''_{t-1} \cdot dt$$

$$y'_t = y'_{t-1} + y''_{t-1} \cdot dt.$$

For width and height, constant growth rate within dt was assumed instead, thus

$$w_t = w_{t-1} + w'_{t-1} \cdot dt$$

$$h_t = h_{t-1} + h'_{t-1} \cdot dt.$$

After prediction, the estimated state $\widehat{\boldsymbol{x}}_t^-$ is corrected by an t observed measurement $\boldsymbol{z}$ at each time step by the steps below:

$$\boldsymbol{K}_t = \boldsymbol{P}_t^-\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_t^-\boldsymbol{H}^T + \boldsymbol{R})^{-1}$$

$$\widehat{\boldsymbol{x}}_t = \widehat{\boldsymbol{x}}_t^- + \boldsymbol{K}_t(\boldsymbol{z} - \boldsymbol{H}\widehat{\boldsymbol{x}}_t^-)$$

$$\boldsymbol{P}_t = (\boldsymbol{1} - \boldsymbol{K}_t\boldsymbol{H})\boldsymbol{P}_t^-.$$

In this case, there was a 10-dimensional measurement vector:

$\boldsymbol{z} = [x^{bg}, y^{bg}, w^{bg}, h^{bg}, x^{det}, y^{det}, w^{det}, h^{det}, v_x, v_y]$, representing the top-left coordinates (x, y), width (w), height (h), reported by background subtraction bg and detector det, separately, as well as the velocity (v_x, v_y) reported by optical flow estimation. The measurement model $\boldsymbol{H}$ is a 10-by-10 matrix of zeros except for ones at $(0,0)$, $(1,1)$, $(2,2)$, $(3,3)$, $(0,4)$, $(1,5)$, $(2,6)$, $(3,7)$, $(4,8)$, $(5,9)$, signifying that the background and detector boxes directly measure x, y, w, and h, and that optical flow directly measures x' and y'. In other words, there are no direct measurements of w', h', x'' or y'' since they are not observable. $\boldsymbol{R}$ is the measurement noise covariance, indicating the noisiness of measurement. By manipulating the measurement noise covariance $\boldsymbol{R}$, we indirectly adjust the Kalman gain $\boldsymbol{K}_t$, indicating the weight of the measurement to update the corresponding prediction.

2.2.2 Measurement Acquisition

It was observed that none of the input measurements such as background subtraction, object detection, and optical flow, corresponded directly to an individually tracked object. Instead, they generated boxes and flow indications for an entire frame. To produce input to an individually tracked object's filter, one should compute a matching of input data to tracked objects. Then apply the matched boxes and flow to the Kalman filter state of the corresponding object.

2.2.2.1 Background Subtraction and Object Detection

Both background subtraction and object detection generate bounding boxes $\{x, y, w, h\}$. We only take those boxes that are consistent with tracker's current state as the measurement. For each Kalman filter, the best matching box as the measurement maximizes the total overlap between the predicted bounds of internal filter and the bounds of the measured boxes. Boxes must overlap with the predicted state in order to be considered a match. Any remaining boxes were used to initialize newly tracked objects.

2.2.2.2 Optical Flow

Optical flow reports velocity on a pixel-by-pixel basis with many erroneous flow vectors, due to the aperture and turbulence problems described earlier. To produce a velocity measurement from the optical flow field for a single object, the noise was first removed from the flow field by forward-backward error thresholding. Then the flow was filtered by location, velocity magnitude, and direction. In particular, consideration was only given to flow vectors that: 1) Originate from the location of the object in the previous frame; 2) Follow the similar direction (within in 45°) of the previous state in order to reduce the effect of the turbulence problem; and 3) Only keep those vectors that fall within twice the error covariance (available in $\boldsymbol{P}$, diagonal entries corresponding to the object velocity in $\hat{\boldsymbol{x}}$), using the simplifying assumption that flow errors follow a normal distribution. Finally, the mean of the remaining flow vectors was used on horizontal and vertical directions, respectively, as the optical flow measurement for the object.

2.2.3 Model Update

The measurements above were directly plugged into the update equations above. However, they are of different quality. Oftentimes, some of the measurements are missing entirely. For example, the invalid foreground is generated when occlusion happens, when a detector misses a small-sized

object, or when an aperture problem gives zero movements of an object. These problems were addressed by varying the measurement noise covariance accordingly, which in turn caused the Kalman filter to choose a gain $\boldsymbol{K}$ that maximized the quality of the internal state. A large measurement covariance $\boldsymbol{Q}$ would result in a smaller $\boldsymbol{K}$, which would cause the measurement to weigh less in correction. When a measurement was missing, the value already in $\widehat{\boldsymbol{x}}$ was used as a proxy and the covariance was set to 1. Consequently, when none of the three measurements were available, the tracker merely relied on the internal prediction, which is called extrapolation. The tracker could still generate tracking results by the internal linear model during extrapolation. The other two cases in Figure 5 demonstrate when at least one bounding box is available (bound update) or only optical flow is obtained (velocity update).

Error gating was also applied to the background subtraction and object detection measurements. For example, only foreground bounding boxes that were well overlapped with their corresponding tracked objects (more than 30% overlap) were used for the update. Meanwhile, the optical flow has a lower confidence with zero value in both directions, since it is unknown whether the object was still, or the aperture problem happened. In addition, as described before, the three measurement types naturally had different error covariances. Although detector had a higher missing rate, the precision was also high. Therefore, a smaller noise covariance was assigned to the measurements from the detector, than to those from the background model.

2.2.4 Tracker Initialization and Termination

Ideally, a fully automatic computer vision-based tracker followed each vehicle as it entered, traversed and exited the scene. The proposed system initialized new trackers based on unmatched boxes from background subtraction and object detection. This supported the challenging cases described above but can result in many spurious trackers due to the noisy nature of both background subtraction and optical flow. A two-pronged approach was used to limit such spurious results. First, initialization was limited to objects larger than 10×10 pixels. This reduced the number of trackers in flight, without significant negative effect. Cars with a reasonable chance of being captured tend to be larger than that in these videos, except when approaching from or driving toward the vanishing point. Second, trackers were terminated when the object left the frame after no direct observations (boxes or optical flow) were made for 50 frames, also known as extrapolation mode. Any object would have such 50 frames before exit. However, upon termination, tracked objects were validated based on the number of observations and distance traveled. Spurious noise tended to be stationary and short-lived, whereas vehicles typically followed a continuous and long-lived path through the scene. To adapt to the variety of video and objects, distance threshold was dynamically computed by object size and video resolution. One good consequence of such scheme was that many early initialized noises were quickly discarded. This is because there was usually no consistent measurement available for them, while those tiny objects discarded as noises were soon available for future initialization, with better quality. Therefore, real small objects were able to be initialized at the earliest point and survive with a complete trajectory.

2.3 DATASET

To the best of our knowledge, there exists no public traffic surveillance video dataset containing complex real-world interactions and illumination variations. Existing vision datasets were either not applicable to the proposed scenario with different a viewpoint (driver's view) or contained short clips with limited adversarial conditions, scale changes, and illumination variations. Even in the largest dataset collected, only 15 out of 98 videos exceed 1000 frames (33 seconds).

11 representative surveillance videos were collected from across the state, from the local Department of Transportation, and annotated these using VATIC (Vondrick et al. 2013). Each object had its location and extent annotated on every frame, which was used as the ground truth. The average length of each video was five minutes (around 9000 frames), sufficient to cover several traffic signal cycles with real-world vehicle interactions and movement patterns. The videos were divided into two groups: simple low resolution (LowRes), and complex high resolution (HighRes). Figure 6 shows screenshots from this dataset, and Table 1 gives an overview of the dataset, where the rightmost four columns indicate the number of videos reflecting various challenging aspects: occlusion, shadows, distortion, and pedestrians.

Table 1. Dataset Overview. The Second and the Third Columns Show the Resolution and Object Size Range in Pixels, Followed by Number of Videos Under Each Group. The Rightmost Four Columns Show the Number of Videos Reflecting Various Challenging Aspects.

Group	Resolution	Object Size	Number	Occlusion	Shadow	Distortion	Pedestrian
LowRes	342×228	$32 - 44{,}814$	5	3	1	3	0
	320×240	$48 - 25{,}284$	2	2	1	2	0
HighRes	720×576	$84 - 255{,}106$	4	3	0	0	1

Figure 6: Snapshots of videos in our dataset, with various resolution, viewpoint, illumination, vehicle size and interactions.

2.4 INTEGRATION WITH TOPIC MODEL

The insight of integrating the topic model is that good tracking results could provide useful information for objects' movement such as entry/exit, size, and velocity. However, with noisy trajectories, such information may not be reliable. On the other hand, such information could help

the tracker eliminate spurious objects and guide the tracking process. This deadlock could be broken by introducing an external module to learn such information. Inspired by this observation, the self-adaptive tracker was proposed given the scene information learned by the topic model. Figure 7 shows the systematic design of it.

The result of the topic model was a set of distributions over the entire scene vocabulary, one for each topic. At a certain frame, each cell in the scene could only move to the most possible (one) direction. As described before, a rough estimation of the entry and exit area of each topic was available. The direction and entry area size could be useful for eliminating spurious objects for initialization. Similarly, trackers were stopped once the object enters the exit area, before getting lost or shifting to other objects. Along the tracking process, although topic model could not provide the detailed information of how much the object should go at the next frame, the object movement is roughly constrained by the most likely direction for the current direction. By continuously checking how much the trajectory aligns with the topic distribution numerically, one may remove the lost tracker along the way and get a score for a naturally terminated tracker. The bounding boxes were used during the object's lifetime to update the statistics in the area, such as size and velocity in each cell. Once enough global statistics are collected, they will be able to help improve the estimation of the tracking.

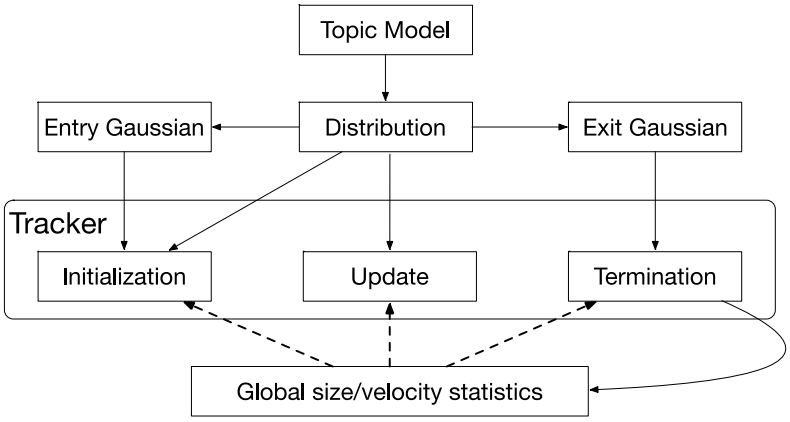

Figure 7: Tracker integrated with topic model.

2.4.1 Topic Match for a Frame

The first step in integrating the topic model was to determine the corresponding topic for the current motions. A frame window of 30 frames was maintained and kept in a visual doc format as described in Chapter 1. Then the current topic z with the maximal the likelihood for the current snippet of the video was used:

$$z^* = \underset{z}{\operatorname{argmax}} \prod_i^V p_{zi}^{n_i} = \underset{z}{\operatorname{argmax}} \left(\log \sum_1^V n_i \cdot p_{zi} \right)$$

p_{zi} was the value of the multinomial at word i in the vocabulary, n_i was the frequency of the word i in the current frame window, V is the vocabulary size. The above computation could be done by matrix multiplication. We implement it with GPU, which does not cause significant computation overhead even though the computation is operated frame wise.

2.4.1.1 Tracker Initialization

At first, with only a rough entry area learned from the topic model, strict filtering was not posed on object initialization. Instead, only boxes with a significant difference in size or distant were filtered. In addition to size, the flow inside the bounding box was compared to the direction indicated by the topic and used for the filtering out of those with different flows. This will be potentially useful for tiny bounding boxes with small movements, where optical flow results tend to be noisy. For the bounding boxes near the entry area with a consistent direction, it was only counted as a good trajectory after evaluation upon termination. When the global statistics have been accumulated for a while, a restricted sizing rule for objects were gradually applied.

2.4.1.2 Tracker Termination

Similar to initialization, the size information for the exit area was not very accurate at the beginning. Apart from the old rules, objects near the exit area were given a higher confidence of leaving. Upon exiting, the object trajectory could be formulated as a very sparse visual document, where the frequency of the word corresponding to the cell on a certain direction is increased once the cell is covered by one bounding box along the history. Then the likelihood is computed as shown in the Topic Match in Section 2.4.1, which produces a numeric score for a trajectory.

2.4.1.3 Global Statistics Update

For a well-tracked object with a higher score, the global statistics with the entire history of the object was updated. Each topic had a separate set of global statistics. A Gaussian for width, height, velocity, was modeled separately on each cell. Its size and velocity statistics were updated once it was covered by one bounding box in the object trajectory.

2.4.1.4 Global Statistics for Tracking

The global statistics generated an additional estimate of where the object would move and how large it will be in the next frame. It was used as another observation for the Kalman filter and updated without changing the covariance matrix. In other words, the observation vector was:

$$\boldsymbol{z} = [x^{bg}, y^{bg}, w^{bg}, h^{bg}, x^{det}, y^{det}, w^{det}, h^{det}, v_x, v_{\boldsymbol{y}}, w^{gs}, h^{gs}, v_x^{gs}, v_{\boldsymbol{y}}^{gs}]$$

gs stands for "global statistics", (w^{gs}, h^{gs}) and $(v_x^{gs}, v_{\boldsymbol{y}}^{gs})$ are size and velocity information, separately. Corresponding change was also applied to the measurement model $\boldsymbol{H}$. Global statistics was particularly useful for objects under projection, where constant velocity/acceleration model could not capture the change rate of the size and velocity. Besides, the topic match equation from Section 2.4.1 was used to check whether or not the object is well tracked. If it does not follow the motion of the current topic, it was discarded as a lost tracker.

CHAPTER 3: VEHICLE COUNTING

Over time, a number of vehicle counting methods have been developed. Some of them include specialized hand-held counting boards with buttons to push, pressure tubes laid across the pavement, magnetic loops under the pavement, and more. Overall, the most powerful techniques rely on manual input and tend to be extremely labor intensive. On the other hand, the mostly automatic techniques lack in accuracy and descriptiveness. In principle, computer vision provides the most scalable and economical alternative. The research team developed a vehicle counter system based on the computer vision tracking algorithm.

3.1 ALGORITHM

The proposed vehicle counter took trajectories generated by the tracker as the input. It required several possible routes, which were call "templates". The trajectories were matched to the most consistent template and the count for the matched template was increased. Suppose there was a set of n templates

$$\boldsymbol{T} = \{T_1, T_2, \dots, T_n\} \text{ and a set of trajectories of } N \text{ objects}$$

$$\boldsymbol{O} = \{O_1^1, O_1^2, \dots, O_1^{t_1}, O_2^1, O_2^2, \dots, O_2^{t_2}, \dots, O_N^1, O_N^2, \dots, O_N^{t_N}\},$$

where t_i is the lifetime of object i. For each object i, the maximal distance to each template is compute, and match the trajectory to the template with minimal maximal distance.

$$T_i^* = \underset{T}{\operatorname{argmin}} \underset{t \in \{1,\dots,t_i\}}{\operatorname{avg}} D(O_i^t, T).$$

The distance measurement was carefully designed, taking both Euclidian distance and direction into consideration. The template was in the form of l line segments $T_m = \{s_{m1}, s_{m2}, \dots, s_{ml}\}$, the distance of a point to a template was considered the closest distance to the segments of the template:

$$D(O_i^t, T_m) = \min_{s \in T_m} d(O_i^t, s) \cdot e^{-\alpha \cdot \cos(p(O_i^t, s))},$$

where $d(O_i^t, s)$ was the perpendicular distance of a point to the line s, $p(O_i^t, s)$ was the angle of the line segment s with the moving direction of object O_i at time t, α was a constant, currently set as 1.

To further ensure that the distance from a trajectory point to the template was the distance to the right template segment, the order of their closest segment was considered. That is, the closest template segment of a point should not be ahead of the closest segments of its previous points. To ensure this property, s^{ij} was defined as the closest segment of jth point of object i. The distance of object O_i to template T_m was computed as follows:

- Find the point $O_i^{t^*}$ with the minimal $D\left(O_i^{t^*}, T_m\right)$.

- For the points before $O_i^{t^*}$: $O_i^t \in \{O_i^1, O_i^2, \dots, O_i^{t^*-1}\}$, the closest segment cannot go beyond the first to its successor's closest segment:
  $D(O_i^t, T_m) = \min_{s \in \{s_{m1}, s_{m2}, \dots, s^{it+1}\}} d(O_i^t, s) \cdot e^{-\alpha \cdot \cos(p(O_i^t, s))}$.
- For the points after $O_i^{t^*}$: $O_i^t \in \{O_i^{t^*+1}, O_i^{t^*+2}, \dots, O_i^{t_i}\}$, the closest segment cannot go beyond its predecessor's closest segment to the last segment:
  $D(O_i^t, T_m) = \min_{s \in \{s^{it-1}, \dots, s_{ml}\}} d(O_i^t, s) \cdot e^{-\alpha \cdot \cos(p(O_i^t, s))}$.

By defining the distance function D with both spatial distance and direction consistency, and keeping closest segment order, one can eliminate those close segments with different directions. Finally, the number of trajectories belonging to each template is the desired vehicle count.

3.2 INTEGRATION WITH TOPIC MODEL

Previously, manually drawn templates to the counter application were provided. This could be tedious for the human. Moreover, the human may have to watch a video for long time to catch all the possible movements in the scene or draw bad lines based on their own understanding. Since the templates are key to the final results, it is necessary to automatically generate them. Fortunately, given the centerlines extracted from topic model results, the research team was able to reduce the centerline into several connecting points and convert it to the template format. Therefore, automatic template generation was achieved.

3.3 USER INTERFACE

A web interface was developed to do the counting. With the integration of the topic model, users only needed to upload the videos, specify the location and the name of each camera, and edit the templates. After some backend processing, the counting results were available to download. The waiting time was also reduced by parallel processing and GPU computation. Figures 8 and 9 show the screenshots of the system.

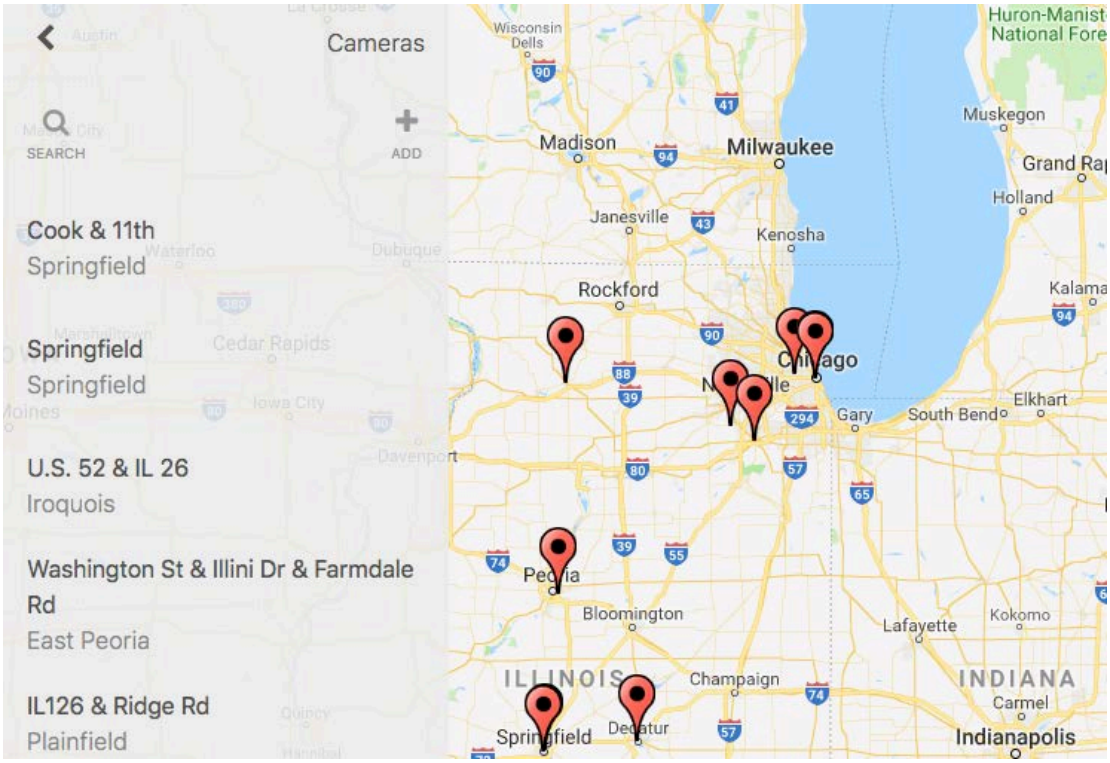

Figure 8: Screenshot of the web portal, cameras are displayed on map.

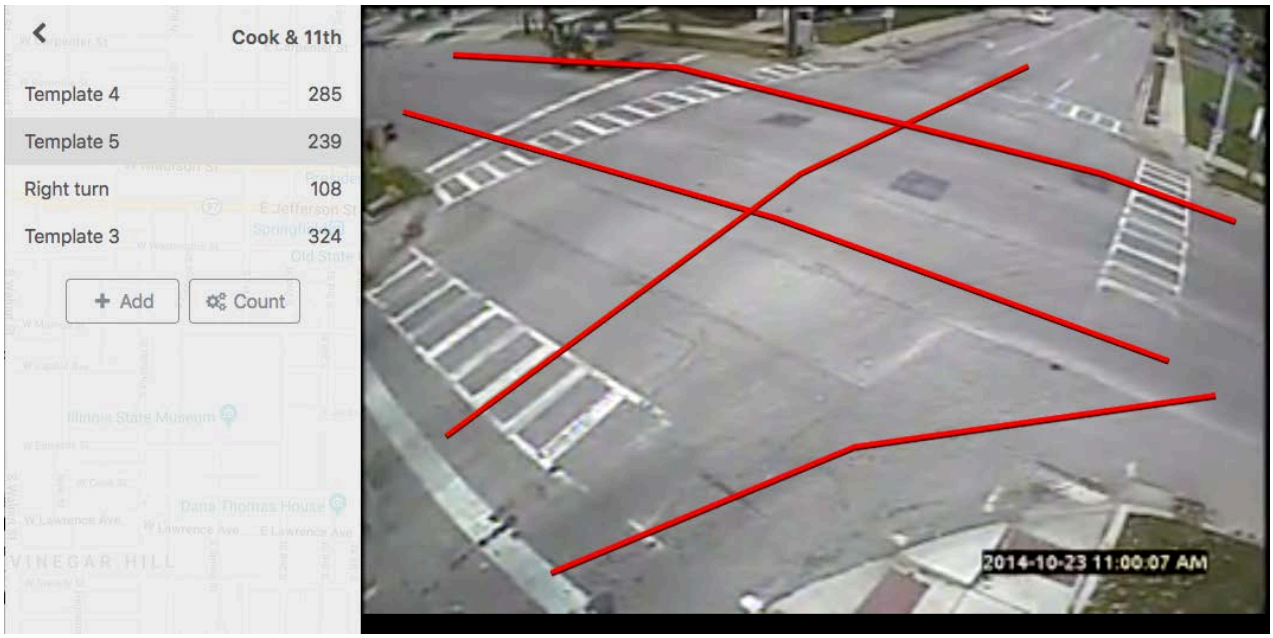

Figure 9: Screenshot of the web portal, left panel shows the counts of each template.

REFERENCES

Barnich, O., and Van Droogenbroeck, M. 2011. “Vibe: A Universal Background Subtraction Algorithm for Video Sequences,” Image Processing, IEEE Trans. On Image Processing, vol. 20, no. 6, pp. 1709–1724.

Ren, S., He, K., Girshick, R., and Sun J. 2017. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” IEEE Transactions on Pattern Analysis and Machine Intelligence vol. 39, no. 6, pp. 1137–1149.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. 2012. “Hierarchical Dirichlet Processes.” Journal of the American Statistical Association vol. 101, no. 476, pp. 1566-1581.

Vondrick, C., Patterson, D., and Ramanan, D. 2013. “Efficiently Scaling up Crowdsourced Video Annotation,” International Journal of Computer Vision, vol. 101, no. 1, pp. 184–204.

Wang, X., Ma, X., and Grimson, W.E.L. 2009. “Unsupervised Activity Perception in Crowded and Complicated Scenes Using Hierarchical Bayesian Models.” IEEE Transactions on Pattern Analysis and Machine Intelligence vol. 31, no. 3, pp. 539–555.

Wu, Y., Lim, J., and Yang, M. H. “Online object tracking: A benchmark,” in *IEEE CVPR*, June 2013.

Xu, H., Zhou, Y., Lin, W., and Zha, H. 2015. “Unsupervised Trajectory Clustering via Adaptive Multi-Kernel-Based Shrinkage,” pp. 4328–4336. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, December 7-13, 2015

Zhao, R., and Wang, X. 2013. “Counting vehicles from semantic regions.” IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 2, pp. 1016–1022.


Illinois Department
of Transportation


ILLINOIS