Traffic Speed Estimation from Surveillance Video Data

For the 2nd NVIDIA AI City Challenge Track 1

Tingting Huang
Institute for Transportation, Iowa State University
2711 S. Loop Dr., Suite 4700, Ames, IA
thuang1@iastate.edu

Abstract

Estimating traffic flow condition is a tough but beneficial task. In Intelligent Transportation System (ITS), many applications have been done to collect and analyze traffic data. However, the surveillance video data are still only used for engineer's manual check. To better utilize this data source, traffic flow estimation from surveillance camera should be explored. This study uses Faster Regional Convolutional Neural Network (Faster R-CNN) with ResNet 101 as the backbone to achieve multi-object detection. Then a tracking algorithm based on histogram comparison is applied to link objects across frames. Finally, this study uses warping method to convert vehicle speeds from the pixel domain to the real world. The results show that estimating vehicle speed at intersection is more challenging than in uninterrupted flow.

1. Introduction

In Intelligent Transportation System (ITS), traffic data is the key to conduct research and applications. Current techniques have made lots of data available for traffic flow monitoring, including using roadside radar sensors on highways, inductive loop detectors at intersections, GPS data collected from probe fleet, etc. Besides those devices, surveillance cameras are deployed widely but have not been used as a mean of traffic data collection yet. Turning the usage of camera from current manual check only to automatic monitoring would be a beneficial challenge. Agencies could not only use surveillance camera as visual validation of incidents, but also as speed data collection and traffic flow monitoring. Thus, this study focuses on estimating vehicle speeds from video data, starting from multi-object detection, tracking to speed conversion, which could demonstrate the potential of turning a camera into a sensor.

To estimate the vehicle speed, the first task is to correctly detect the vehicles showing in the video. The object detection has been studied with computer vision techniques for years. Traditional computer vision techniques tend to analyze the object's contour, contrast and other features to detect the object. However, with the successful implementation of artificial intelligence technology, especially the deep convolutional neural network (CNN), the object detection has been improved. The first deep CNN used for image classification was developed by Alex Krizhevsky [1] in ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2012). Known as AlexNet, it is deeper and has more filters than previous LeNet [2]. It outperformed all other methods in the challenge with a large margin. After the success of AlexNet, more CNN variants (ZFNet [3], GoogLeNet [4], VGGNet [5], ResNet [6]) are developed in object detection domain.

Recently, researchers also found that using regional proposals before using CNN to extract features could improve the performance on object detection [7]. To overcome the long computational time of R-CNN, several improvements have also been made. Fast R-CNN [8] and Faster R-CNN [9] are developed to speed up the computation. This study uses Faster R-CNN with ResNet 101 model to achieve multiple vehicle detection.

The next for speed estimation is tracking. Multi-object tracking (MOT) is an important task in video processing. Many studies have been done in both batch and online manner. And many applications have also been made to implement MOT. For example, OpenCV provides different kinds of tracker from different algorithm, such as, MIL, BOOSTING, TLD, KCF, etc. [10]. These methods have their own advantages and disadvantages. For this specific challenge, I propose a histogram-based tracking algorithm to solve the MOT in traffic estimation. Details will be discussed in section 3.2.

The last step for traffic speed estimation is converting the speed obtained from the pixel domain to the real world. Camera calibration is a tough task in image processing, especially when there is no enough meta information about the camera, such as in this challenge. Image warping is a way to convert the perspective from one way to another. Due to the limitation of time in this challenge, a warping method by linear perspective transformation was, with each testing location manually calibrated.

2. Data description

The test data for Challenge Track 1 is collected by challenge organizers from 2 highway interchanges and 2 signalized intersections located in urban area in state of California. The locations and some characteristics are summarized below.

Number	Type	Description	Coordinates
1	Highway	I-280 @ Winchester	37.316788, -121.950242
2	Highway	I-280 @ Wolfe	37.330574, -122.014273
3	Intersection	San Tomas @ Saratoga	37.326776, -121.965343
4	Intersection	Stevens Creek @ Winchester	37.323140, -121.950852

Table 1. Location information of data collected

Each location has multiple video clips, and each clip lasts 1 minute long with 30 fps. The resolution is 1920×1080 for each video. All the cameras have overhead view with no additional information on mounting height and angle.

3. Methods and results

The approach to traffic speed estimation in this challenge is outlined in Figure 1. Details in different phases are discussed as follows.

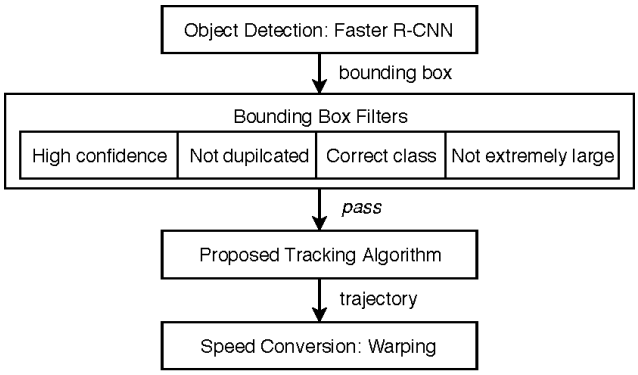


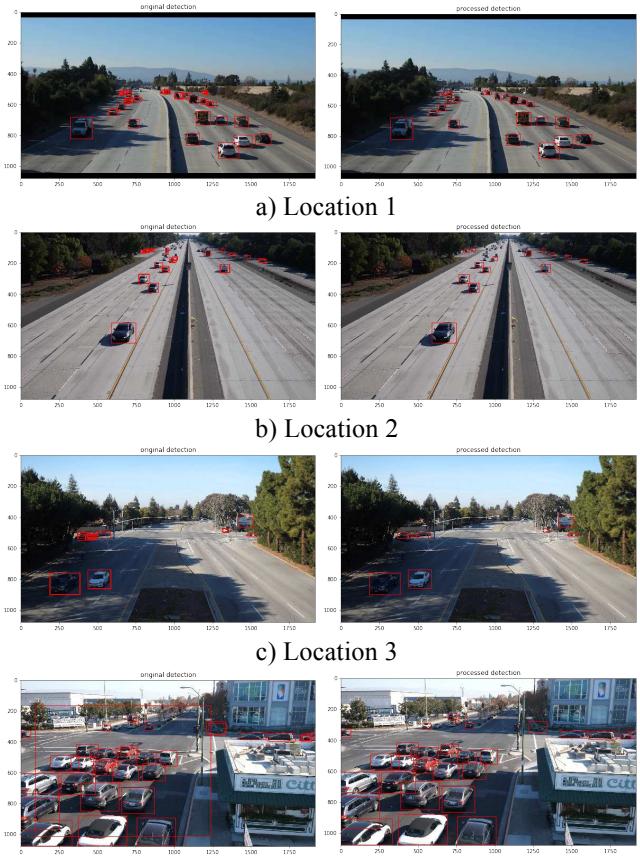
Figure 1. Speed estimation pipeline.

3.1. Object detection

Due to the lack of training data in this challenge, a pre-trained model has been selected for object detection. By investigating the performance of the pre-trained models from TensorFlow object detection API [11][13], the Faster R-CNN with ResNet 101 as the backbone trained on COCO dataset [12] provides a fair inference speed and relatively high performance on the challenge dataset. Among 80 classes in COCO, car, bus and truck are the object of interest in this study.

All frames are extracted from the 27 evaluation videos and the inference frozen graph (2017-11-08 version) is then applied to detect objects in each frame. The inference speed is about one frame per second on one NVIDIA TITAN Xp GPU. The false positives in the detected bounding boxes were reduced by a group filtering algorithms: a) class filter, b) confidence score filter, c) duplicated detection filter, and d) outlier filter. As mentioned above, only class 3, 6 and 8 in COCO are vehicles. Thus, class filter will only remain the results from those classes. Confidence scores are generated during detection process, in this challenge, 0.2 is used for maintain enough amount of detections. For duplicated detections, intersection-over-union (IoU) over 0.3 is used for identifying overlapping duplicates. And extremely large bounding box is also removed. These thresholds are configurable inputs subject to different use cases.

After applying those filtering algorithms, a sample frame with the final detection results are shown in Figure 2, comparing to the same frame with the raw detections on the left.



a) Location 1

b) Location 2

c) Location 3

d) Location 4

Figure 2. Raw detection and processed detection for 4 locations.

As shown in Figure 2, the left column illustrates the original detection from Faster R-CNN model. The original detection contains many duplicate detections, especially at the far end of the view (location 1, 2 and 3). Location 4 also has an extreme large detection covered the whole intersection. These kinds of error should be removed from detection results to help the tracking phase later.

3.2. Multi-object tracking

In this challenge, I propose a histogram-based tracking algorithm. The histogram-based tracker connects the same object across frames by finding the minimum Chi-squared distance (calculated in OpenCV [14]) in the histogram domain among a group of candidates. The algorithm is descripted as follows.

Tracking Algorithm

Input frames with bounding boxes, F
$i = 0$
while $i \leq N - 1$ **do**
  for t in $F[i]$ **do**
    Generate neighborhood region r: extend t by its height and width on $F[i+1]$.
    for b in $F[i+1]$ **do**
      if $inRate(b,\ r) > 0.8$ **then**
        b appends to C
      end if
    end for
    Compare target image (t) with candidate images (C).
    Select image m in C with minimum distance.
    Assign ID on $C[m]$ same as t.
  end for
  for b in $F[i+1]$ **do**
    if b is not labeled **then**
      Assign b a global ID.
    end if
  end for
  i+=1
end while

Input all N frames $F[\cdot]$ to the tracker, and objects in consecutive frames $F[i]$ and $F[i+1]$ are compared. The tracking algorithm is explained in details as the following.

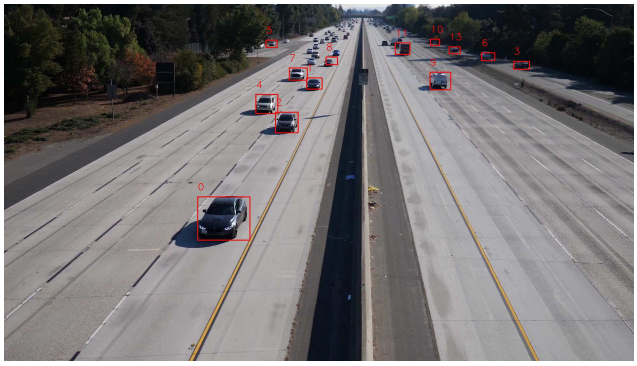
- **Select candidates in the next frame**

In this study the videos are recorded at 30fps so that an assumption was made that the same object in the next frame will stay in the neighborhood region of its current bounding box position. Specifically, the neighborhood region r in the next frame $F[i+1]$ is a squared region at the same location of the object's current bounding box t in the current frame and with weight and height three times as large. All bounding boxes b with 80% area covered by the neighborhood region r in the next frame $F[i+1]$ are selected as candidates C with respect to the target object t. This conservative candidate selecting strategy is supposed to reduce the broken trajectories to a fairly good level.

- **Find the minimum distance**

After experimenting on several similarity comparison methods, the Chi-squared distance in histogram domain was selected due to its fairly high variation and distinguishability among the vehicles in the study data set and the simplicity in implementation. The candidate $C[m]$ with the lowest Chi-squared distance from the target object t is labeled with the same object ID. The same method is applied to assign ID to all bounding boxes b in the next frame $F[i+1]$ and all the remaining unassigned bounding boxes are treated as new objects with new IDs.

The proposed tracking algorithm runs at over 30 frames per second. A tracking example is shown in Figure 3 using frame 1 and frame 3 from location 2 to demonstrate the movement of vehicles. At the bottom right of Figure 3b, a new vehicle showed up and assigned ID 14. Other vehicles were tracked with correct ID assigned except object 11 was missed due to the loss of detection.



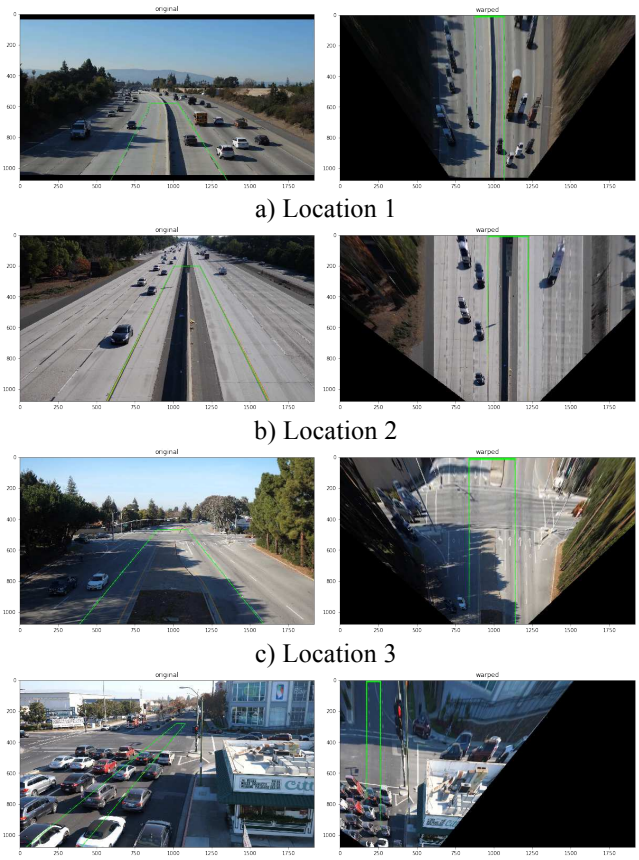
a) Frame 1

b) Frame 3

Figure 3. Tracking example from location 2.

3.3. Speed conversion

The method to convert the speed from pixel per second to mile per hour used here is warping with linear perspective transformation. Examples of warped images from 4 locations are shown in Figure 4. Due to the different camera configurations across videos even at the same location, each video was calibrated separately.



a) Location 1

b) Location 2

c) Location 3

d) Location 4

Figure 4. Image warping samples from each video.

Using the assumption that the lane width is 12 feet (US standard), the pixel distance of warped lane is used as the reference to convert the speed. The raw speed conversion results are smoothed by moving average with step size of 5 frames (0.167 seconds).

The speed conversion is the key step and can potentially be the major source of error in the entire speed estimation pipeline. The warping method with linear perspective transformation may introduce error due to its strong assumptions, such as the road is straight and flat, no camera distortion, and detected centroids of vehicles (the bottom midpoint of a bounding box is used as the centroids) are on the ground. These assumptions are not necessarily met in this challenge, thus, a better pixel-to-reality calibration method should be considered in future study. In this study, an alternative approach is used to process the noisy raw data from the linear perspective transformation.

For highway locations that are usually in free flow condition with rare stop-and-go pattern, the assumption that drivers are driving around speed limit is made. The detected speed from the aforementioned linear perspective transformation can provide the essential speed variation among vehicles, and a scaling method is applied so that the final detected speeds comply to the speed limit while the detected variations are remained. The final results on highway locations have the RMSE as low as 3.9083.

However, the intersection always has stop-and-go pattern, thus, no further process except smoothing is used. The final combined results on both highway and intersection locations have the RMSE of 8.6089 and total detection rate (DR) of 0.8148.

4. Conclusion

Traffic speed estimation is important in many aspects of traffic operation and management, such as flow monitoring, incident detection, and delay cost estimation, etc. Current data collection methods like radar sensor or inductive loop detector are costly and surveillance cameras are only used for manual check now. Turning low-cost cameras into effective sensors is a beneficial challenge. With computer vision and deep learning techniques rapidly developed, this study tries to extract vehicle speeds from surveillance video data.

This study aims to solve the 2018 AI City Challenge Track 1. Three steps are taken: a) multi-object detection using Faster R-CNN, b) multi-object tracking based on histogram comparison, and c) speed conversion using warping with linear perspective transformation. The detection and tracking yield plausible results but the warping with linear perspective transformation introduces errors due to its strong linear assumptions. To overcome the limitation in warping process, this study applies a scaling method based on assumption that highway traffic is under free flow condition. The results show that by using this approach, the speed estimation on highway achieved a good performance. However, this approach has strict assumption and low scalability, which should be improved in future study.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet classification with deep convolutional neural networks, Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, vol. 86, no.11, pp. 2278–2324, 1998.

[3] M. D. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, arXiv:1311.2901 [cs], Nov. 2013.

[4] C. Szegedy et al., Going Deeper with Convolutions, arXiv:1409.4842 [cs], Sep. 2014.

[5] K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556 [cs], Sep. 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, arXiv:1512.03385 [cs], Dec. 2015.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, arXiv:1311.2524 [cs], Nov. 2013.
[8] R. Girshick, Fast R-CNN, arXiv:1504.08083 [cs], Apr. 2015.
[9] S. Ren, K. He, R. Girshick, and J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, arXiv:1506.01497 [cs], Jun. 2015.
[10] OpenCV. Introduction to OpenCV Tracker. [Online]. Available: https://docs.opencv.org/3.1.0/d2/d0a/tutorial_introduction_to_tracker.html. [Accessed: 04-Apr-2018].
[11] Google. Models: Models and examples built with TensorFlow. [Online]. Available: https://github.com/tensorflow/models. [Accessed: 04-Apr-2018].
[12] COCO - Common Objects in Context. [Online]. Available: http://cocodataset.org/#home. [Accessed: 04-Apr-2018].
[13] Google. TensorFlow detection model zoo. [Online]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md. [Accessed: 04-Apr-2018]
[14] OpenCV. Histogram Comparison - OpenCV 2.4.13.6 documentation. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html. [Accessed: 04-Apr-2018].