Real-Time Work Zone Traffic Management via Unmanned Air Vehicles

Project No. 19ITSOSU01
Lead University: Louisiana State University
Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation’s University Transportation Centers Program. However, the U.S. Government assumes no liability for the contents or use thereof.

Acknowledgements

We would like to express our special thanks to the Project Review Committee as well as the director of Tran-SET Dr. Marwa Hassan, who gave us support and direction during the whole project.
Real-Time Work Zone Traffic Management via Unmanned Air Vehicles

Dr. Charles Malveaux (PI), ORCID 0000-0001-7903-4622
Dr. Marcio de Queiroz (Co-PI), ORCID 0000-0003-3878-4392
Dr. Xin Li (Co-PI), ORCID 0000-0002-0144-9489
Dr. Hany Hassan (Co-PI), ORCID 0000-0002-3255-7474
Dr. Zewei He (Post Doc), ORCID 0000-0003-4280-9708

Transportation Consortium of South-Central States (Tran-SET)
University Transportation Center for Region 6
3319 Patrick F. Taylor Hall, Louisiana State University, Baton Rouge, LA 70803

United States of America
Department of Transportation
Research and Innovative Technology Administration

No restrictions. This document is available through the National Technical Information Service, Springfield, VA 22161.
### APPROXIMATE CONVERSIONS TO SI UNITS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>When You Know</th>
<th>Multiply By</th>
<th>To Find</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>LENGTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>in</td>
<td>inches</td>
<td>25.4</td>
<td>millimeters</td>
<td>mm</td>
</tr>
<tr>
<td>ft</td>
<td>feet</td>
<td>0.305</td>
<td>meters</td>
<td>m</td>
</tr>
<tr>
<td>yd</td>
<td>yards</td>
<td>0.914</td>
<td>meters</td>
<td>m</td>
</tr>
<tr>
<td>mi</td>
<td>miles</td>
<td>1.61</td>
<td>kilometers</td>
<td>km</td>
</tr>
<tr>
<td>in²</td>
<td>square inches</td>
<td>645.2</td>
<td>square millimeters</td>
<td>mm²</td>
</tr>
<tr>
<td>ft²</td>
<td>square feet</td>
<td>0.093</td>
<td>square meters</td>
<td>m²</td>
</tr>
<tr>
<td>yd²</td>
<td>square yard</td>
<td>0.836</td>
<td>square meters</td>
<td>m²</td>
</tr>
<tr>
<td>ac</td>
<td>acres</td>
<td>0.405</td>
<td>hectares</td>
<td>ha</td>
</tr>
<tr>
<td>mi²</td>
<td>square miles</td>
<td>2.59</td>
<td>square kilometers</td>
<td>km²</td>
</tr>
<tr>
<td>VOLUME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fl oz</td>
<td>fluid ounces</td>
<td>29.57</td>
<td>milliliters</td>
<td>mL</td>
</tr>
<tr>
<td>gal</td>
<td>gallons</td>
<td>3.785</td>
<td>liters</td>
<td>L</td>
</tr>
<tr>
<td>ft³</td>
<td>cubic feet</td>
<td>0.028</td>
<td>cubic meters</td>
<td>m³</td>
</tr>
<tr>
<td>yd³</td>
<td>cubic yards</td>
<td>0.765</td>
<td>cubic meters</td>
<td>m³</td>
</tr>
<tr>
<td><strong>NOTE:</strong> volumes greater than 1000 L shall be shown in m³</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>oz</td>
<td>ounces</td>
<td>28.35</td>
<td>grams</td>
<td>g</td>
</tr>
<tr>
<td>lb</td>
<td>pounds</td>
<td>0.454</td>
<td>kilograms</td>
<td>kg</td>
</tr>
<tr>
<td>T</td>
<td>short tons (2000 lb)</td>
<td>0.907</td>
<td>megagrams (or &quot;metric ton&quot;)</td>
<td>Mg (or &quot;t&quot;)</td>
</tr>
<tr>
<td>TEMPERATURE (exact degrees)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>°F</td>
<td>Fahrenheit</td>
<td>5 (°F-32)/9</td>
<td>Celsius</td>
<td>°C</td>
</tr>
<tr>
<td>ILLUMINATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fc</td>
<td>foot-candles</td>
<td>10.76</td>
<td>lux</td>
<td>lx</td>
</tr>
<tr>
<td>fl</td>
<td>foot-Lamberts</td>
<td>3.426</td>
<td>candela/m²</td>
<td>cd/m²</td>
</tr>
<tr>
<td>FORCE and PRESSURE or STRESS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lbf</td>
<td>poundforce</td>
<td>4.45</td>
<td>newtons</td>
<td>N</td>
</tr>
<tr>
<td>lbf/in²</td>
<td>poundforce per square inch</td>
<td>6.89</td>
<td>kilopascals</td>
<td>kPa</td>
</tr>
</tbody>
</table>

### APPROXIMATE CONVERSIONS FROM SI UNITS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>When You Know</th>
<th>Multiply By</th>
<th>To Find</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>LENGTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mm</td>
<td>millimeters</td>
<td>0.039</td>
<td>inches</td>
<td>in</td>
</tr>
<tr>
<td>m</td>
<td>meters</td>
<td>3.28</td>
<td>feet</td>
<td>ft</td>
</tr>
<tr>
<td>m</td>
<td>meters</td>
<td>1.09</td>
<td>yards</td>
<td>yd</td>
</tr>
<tr>
<td>km</td>
<td>kilometers</td>
<td>0.621</td>
<td>miles</td>
<td>mi</td>
</tr>
<tr>
<td>mm²</td>
<td>square millimeters</td>
<td>0.0016</td>
<td>square inches</td>
<td>in²</td>
</tr>
<tr>
<td>m²</td>
<td>square meters</td>
<td>10.764</td>
<td>square feet</td>
<td>ft²</td>
</tr>
<tr>
<td>m²</td>
<td>square meters</td>
<td>1.195</td>
<td>square yards</td>
<td>yd²</td>
</tr>
<tr>
<td>ha</td>
<td>hectares</td>
<td>2.47</td>
<td>acres</td>
<td>ac</td>
</tr>
<tr>
<td>km²</td>
<td>square kilometers</td>
<td>0.386</td>
<td>square miles</td>
<td>mi²</td>
</tr>
<tr>
<td>VOLUME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mL</td>
<td>milliliters</td>
<td>0.034</td>
<td>fluid ounces</td>
<td>fl oz</td>
</tr>
<tr>
<td>L</td>
<td>liters</td>
<td>0.264</td>
<td>gallons</td>
<td>gal</td>
</tr>
<tr>
<td>m³</td>
<td>cubic meters</td>
<td>35.314</td>
<td>cubic feet</td>
<td>ft³</td>
</tr>
<tr>
<td>m³</td>
<td>cubic meters</td>
<td>1.307</td>
<td>cubic yards</td>
<td>yd³</td>
</tr>
<tr>
<td>MASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>grams</td>
<td>0.035</td>
<td>ounces</td>
<td>oz</td>
</tr>
<tr>
<td>kg</td>
<td>kilograms</td>
<td>2.202</td>
<td>pounds</td>
<td>lb</td>
</tr>
<tr>
<td>Mg (or &quot;t&quot;)</td>
<td>megagrams (or &quot;metric ton&quot;)</td>
<td>1.103</td>
<td>short tons (2000 lb)</td>
<td>T</td>
</tr>
<tr>
<td>TEMPERATURE (exact degrees)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>°C</td>
<td>Celsius</td>
<td>1.8C+32</td>
<td>Fahrenheit</td>
<td>°F</td>
</tr>
<tr>
<td>ILLUMINATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lx</td>
<td>lux</td>
<td>0.0929</td>
<td>foot-candles</td>
<td>fc</td>
</tr>
<tr>
<td>cd/m²</td>
<td>candela/m²</td>
<td>0.2919</td>
<td>foot-Lamberts</td>
<td>fl</td>
</tr>
<tr>
<td>FORCE and PRESSURE or STRESS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>newtons</td>
<td>0.225</td>
<td>poundforce</td>
<td>lbf</td>
</tr>
<tr>
<td>kPa</td>
<td>kilopascals</td>
<td>0.145</td>
<td>poundforce per square inch</td>
<td>lbf/in²</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

TECHNICAL DOCUMENTATION PAGE ................................................................................................................................. ii
LIST OF FIGURES ........................................................................................................................................................................ v
ACRONYMS, ABBREVIATIONS, AND SYMBOLS ............................................................................................................................ vi
EXECUTIVE SUMMARY ................................................................................................................................................................... vii
1. INTRODUCTION ........................................................................................................................................................................ 1
2. OBJECTIVES ................................................................................................................................................................................ 2
3. LITERATURE REVIEW .................................................................................................................................................................. 3
4. METHODOLOGY .......................................................................................................................................................................... 5
   4.1 Vehicle Detection Algorithm: Based on YOLOv3 ....................................................................................................................... 5
   4.2 Vehicle Tracking Algorithm: Based on Deep_SORT .................................................................................................................. 6
   4.3 Speed Estimation Algorithm .................................................................................................................................................... 7
5. ANALYSIS AND FINDINGS .......................................................................................................................................................... 11
   5.1 Output Results .......................................................................................................................................................................... 11
   5.2 Speed Estimation Experimental Results .................................................................................................................................. 12
   5.3 Occlusion ................................................................................................................................................................................... 14
   5.4 Manual Annotation .................................................................................................................................................................. 15
   5.5 Traffic Management App ....................................................................................................................................................... 15
6. CONCLUSIONS ............................................................................................................................................................................ 18
REFERENCES ............................................................................................................................................................................... 19
APPENDIX A: Codes and Configurations ................................................................................................................................. 21
LIST OF FIGURES

Figure 1. System architecture. ........................................................................................................ 2
Figure 2. The original video frame and the ROI. The rectangular denotes the chosen region. ...... 5
Figure 3. YOLOv3 network architecture. ....................................................................................... 6
Figure 4. Output result of detection algorithm ............................................................................. 6
Figure 5. Output result of tracking algorithm ............................................................................... 7
Figure 6. Left: arbitrary view, Right: fronto-parallel view. ............................................................ 8
Figure 7. The output result of rectification. (a) before rectification, (b) after rectification .......... 8
Figure 8. Output result of our algorithms (video1). The resultant videos are annotated with bounding boxes, moving trajectories and speed values ........................................................... 11
Figure 9. Output result of our algorithms (video2). ................................................................... 12
Figure 10. Speed estimation experimental results when the real speed values are 25 mph and 30 mph. Top: 25 mph, Bottom: 30 mph. ....................................................................................... 13
Figure 11. Speed estimation experimental results when the real speed values are 40 mph and 50 mph. Top: 40 mph, Bottom: 50 mph. ....................................................................................... 13
Figure 12. Estimated speed values when the real speed is 25 mph. .......................................... 14
Figure 13. Estimated speed values when the real speed is 50 mph. .......................................... 14
Figure 14. When our algorithms meet occlusion ........................................................................ 15
Figure 15. Manual Annotation on the reference frame .............................................................. 15
Figure 16. App name: INRIX TRAFFIC ..................................................................................... 16
Figure 17. App name: Google maps .............................................................................................. 17
## ACRONYMS, ABBREVIATIONS, AND SYMBOLS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>DJI</td>
<td>Da-Jiang Innovations</td>
</tr>
<tr>
<td>DOT</td>
<td>Department of Transportation</td>
</tr>
<tr>
<td>FAA</td>
<td>Federal Aviation Administration</td>
</tr>
<tr>
<td>I10</td>
<td>interstate highway 10</td>
</tr>
<tr>
<td>ROI</td>
<td>region of interest</td>
</tr>
<tr>
<td>VTOL</td>
<td>vertical take-off and landing</td>
</tr>
<tr>
<td>UAS</td>
<td>unmanned aerial system</td>
</tr>
<tr>
<td>UAV</td>
<td>unmanned aerial vehicle</td>
</tr>
<tr>
<td>YOLO</td>
<td>you only look once</td>
</tr>
</tbody>
</table>
EXECUTIVE SUMMARY

Highway work zones are prone to traffic accidents when congestion and queues develop. Vehicle queues expand at a rate of over 1 mile every 2 minutes. Back-of-queue, rear-end crashes are the most common work zone crash, endangering the safety of motorists, passengers, and construction workers. The dynamic nature of queuing in the proximity of highway work zones necessitates traffic management solutions that can monitor and intervene in real time. Fortunately, recent progress in sensor technology, embedded systems, and wireless communication coupled to lower costs are now enabling the development of real-time, automated, “intelligent” traffic management systems that address this problem.

The goal of this research was to perform preliminary experimentation and proof of concept development work for the use of unmanned aerial system (UAS) to monitor highway construction zone traffic in order to create real-time alerts for motorists, construction workers, and first responders. The main tasks of the proposed system were to collect traffic data via the unmanned aerial vehicle (UAV)’s camera, analyze collected data and to prove whether or not a UAV based highway construction zone monitoring system would be capable of detecting congestion and back-of-queue information, and alerting stake holders including drivers, construction workers, and first responders.

Experiments were conducted using UAS to monitor traffic and collect traffic videos for processing. Prototype software was created to analyze this data. In the process of conducting experiments researchers went to a remote area that had a view of highway traffic and took sample video data of vehicle traffic. This vehicle traffic data was processed using a machine learning algorithm and custom coding. From this data the speed and relative motion of vehicle traffic was detected using the combination of custom software and machine learning algorithms. Matlab was specifically used to process the large video data files generated by research experiments. After preliminary results proved the promise of vehicle speed detection using the algorithm that researchers developed additional research was conducted to calibrate the custom software’s ability to calculate vehicle speed, crucial to reporting on the flow or stoppage of traffic in highway construction zones. Calibration experiments were performed by having one experimenter drive a commercially available vehicle at a set speed on a measured course while other researchers created video. This video data was then processed using custom software and comparisons were made between experimental speed data from video processing and actual vehicle speed as set by the driver.

Results showed that the software was successful in detecting vehicle speed from zero mph to highway speeds. Calibration experiments were able to bring vehicle speed detection from video to within 2 mph of actual vehicle speed. A combination of commercially available machine learning algorithms, Matlab data processing software, and custom coding by researchers was successful in creating a method for processing and analyzing highway traffic flow data.

Vehicle speeds from zero mph to highway speeds were detected. Reviews of available mobile traffic apps were conducted and 2 applications, INRIX and Google Maps were chosen for future integration with advanced iterations of the UAV and software system that has been created by this research. This project has proven that UAS monitoring of highway construction zones and real-time alerts to motorists, construction crews, and first responders is possible in the near term and
future research is needed to further development and implement the innovative UAS traffic monitoring system developed by this research.
1. INTRODUCTION

Data shows that the risk of traffic accidents increases when traffic congestion and queues develop in highway work zone areas. As speeds’ variations increase on highways, the likelihood of traffic crashes rise. Restricted views of the road ahead caused by surface and road features such as hills, and curves along other obstructions such as work zone areas can make it hard for motorists to anticipate traffic slowdowns and react accordingly in order to avoid accidents. This contributes to a high number of back-of-queue, rear-end crashes which are the most common type of work zone crashes. Providing early warnings to motorists, road workers, and emergency personnel to help avoid crashes due to work-related traffic slowdowns is essential to increase highway work zone safety.

Queuing occurs when traffic demand exceeds the capacity of the roadway. Once queuing occurs, the resulting traffic blockage can expand from the site of incidence at a rate of up to 40 miles per hour. This translates to up to 1 mile of added length for every 2 minutes of queuing.

The dynamic and unpredictable nature of highway queuing in the proximity of work zones necessitates traffic management solutions that can monitor and intervene in real time. In such situations, it is difficult to justify constant personnel time devoted to monitoring conditions and then initiating a response when queuing occurs. Fortunately, recent progress in sensor technology, embedded systems, and wireless communication coupled to lower costs are now enabling the investigation of real-time, automated, “intelligent” traffic management systems that address this problem.

This project utilized an unmanned air vehicle (UAV)-based remote sensing system that can help in minimizing highway work zone accidents and improving traffic flow at work zones. The UAV, equipped with an onboard camera, can provide real-time traffic monitoring with high temporal and spatial resolution. Unlike induction loops, overhead radars, and fixed cameras, which are commonly used to monitor traffic and are limited to static point measurements, the UAV provides mobility and range to the monitoring system. We envision the UAV-based system dynamically managing work zone safety and traffic mobility challenges by interacting in real time with changeable message boards, motorist cellphones, traffic management center, and highway patrol personnel.
2. OBJECTIVES

The objectives of our project can be summarized as follows:

- **Develop a prototype UVA-based remote sensing system (see in Figure 1) for highway construction site monitoring.**
  - Drone for capturing videos
    
    There are two kinds of UAVs: (1) multi-rotor UAVs, and (2) fixed-wing UAVs. For this project, the UAVs should have the ability to continuously work for enough time (more than 30 minutes), should have vertical take-off and landing (VTOL) capability and can hover in the air (to provide stable images of the traffic). We actually chose a commercial drone, DJI (Da-Jiang Innovations) Inspiration, which meets the requirements for our project.
  - Server for processing the videos
  - Alert system for sending alerts

- **This system provides traffic conditions data at highway work zones. Researchers explored the deep-learning-based algorithms needed to identify vehicles, track their movement, and analyze their speed. This research is about traffic data processing. Researchers have developed detection, tracking and speed estimation algorithms for this project.**
  - Detection: identify all the vehicles in an image
  - Tracking: track the movement of each detected vehicle
  - Speed estimation: calculate the speed values of each tracked vehicle

- **This data can be used to improve both traffic safety and operations at work zones.**
  - The proposed future system can send traffic alert information via message boards located along side or above the roadway, cellular applications, and highway advisory radio systems.

![Figure 1. System architecture.](image-url)
3. LITERATURE REVIEW

In recent years, the use of technology to improve work zone safety has received considerable attention by some state DOTs. A good source for such efforts is the Smarter Work Zones initiative of the National Work Zone Safety Information Clearinghouse which focuses on “using technology applications to dynamically manage traffic in the work zone environment” (I). Among the technology applications are real-time traveler information, queue warning, dynamic lane merge, incident management, variable speed limits, automated enforcement, and entering/exiting construction vehicle notification. Reports on the system deployment for some of these applications are provided elsewhere (1).

One private business that provides smart work zone solutions is Ver-Mac through their JamLogic system (2). The system is composed of a suite of fixed field sensors (camera, Doppler, microwave) that wirelessly communicate with a computer server that analyzes the data. Information is then sent out to message boards, motorists, traffic management center, and project managers in real time.

Literature review has shown that no UAV-based system has been officially implemented for work zone traffic management. However, such systems have been proposed for general traffic monitoring and analysis. For example, G. Salvoa et al. (3) described a system to evaluate traffic flow conditions in urban areas (Palermo, Italy) based on videos acquired by an UAV. In (4), an UAV system was proposed to monitor abnormal driving behaviors (e.g., driving while intoxicated) thereby, preventing accidents and promoting road safety. Comprehensive surveys of numerous UAV-based systems developed for traffic monitoring can be found in (5) (pre-2005 systems) and (6) (2005-2012 systems). In addition, M.A. Khana et al. (7) presented general guidelines for designing an UAV-based system for traffic applications. J. Lee et al. (8) explored the applicability of small quadcopter drones for traffic surveillance and roadway incident monitoring. E. Barmpounakis and N. Geroliminis (9) designed one-of-a-kind experiment to monitor urban congestion with a swarm of drones, and created the most complete urban multimodal dataset, nicknamed pNEUMA, to study congestion.

The current Federal Aviation Administration (FAA) regulations for the operation of small UAVs (a.k.a. Part 107 rule) include the following rules which are relevant to this project (10):

- UAVs cannot be flown over moving traffic.
- UAV plus payloads must weigh less than 55 lbs.
- Operator must keep UAV within sight or, if using first-person-view, must have an observer with unaided view of the UAV.
- UAV must be flown 30 minutes before official sunrise to 30 minutes after official sunset, local time. In twilight, it can be flown if equipped with anti-collision lighting.
- Maximum allowable altitude is 400 feet above the ground or a structure.
- UAV cannot be flown over anyone not directly involved in its operation nor from a moving vehicle.
- Payloads must be securely attached and should not adversely affect the flight characteristics or the UAV controllability.
- Operator must have a remote pilot certificate with a small UAV rating, or be directly supervised by someone who holds this certificate.
It is important to note that waivers can be requested for certain rules if the petitioner can demonstrate that the UAV operation will not endanger other aircraft, people, or property on the ground or in the air.

In the state of Louisiana, a commercial pilot is required to follow the requirements of FAA’s Part 107 rule (11). In addition, the State Legislature introduced several drone laws between 2014 and 2017 among which is SB 69, which states that only the State may regulate UAV operation, pre-empting local regulation (11).
4. METHODOLOGY

In order to conduct the research, researchers captured video data via a DJI drone near interstate highway 10. After capturing this data, researchers isolated the region of interest (ROI) in the captured video data, i.e., the road area, and discarded the useless area in order to enable data processing, e.g., the sky area. The original video frames and the ROIs are illustrated in Figure 2.

![Image of original video frame and ROI]

Figure 2. The original video frame and the ROI. The rectangular denotes the chosen region.

4.1 Vehicle Detection Algorithm: Based on YOLOv3

The aim of vehicle detection is to identify all the vehicles in an image (12, 13). Our algorithm is based on YOLOv3 (14). It is a one-stage deep learning based detection algorithm, which detects objects in images using a single deep learning neural network. The most important feature of YOLOv3 is its real-time efficiency. It can reach a good tradeoff between speed and accuracy. Recently, YOLOv3 was widely adopted in detection tasks.

The network architecture of YOLOv3 is illustrated in Figure 3. The most salient feature of YOLOv3 is that it makes detections at three different scales. YOLOv3 uses a variant of Darknet, which originally has 53 layers trained on ImageNet. For the task of detection, 53 more layers are stacked onto it, so YOLOv3 has a total 106 layers. As previously stated YOLOv3 makes prediction at three scales: Scale 1 is responsible for detecting large objects; Scale 2 is responsible for detecting medium objects; Scale 3 is responsible for detecting small objects.
The output results of our detection algorithm are illustrated in Figure 4 by researchers. The resultant videos are annotated with bounding boxes. In object detection, we usually use a bounding box to describe the target location. The bounding box is a rectangular box that can be determined by the coordinates of the image in the upper-left corner and the coordinates in the bottom-right corner. YOLOv2 and v3 use anchor points to predict the center of the bounding box, and then estimate the outer edges and create the bounding box as seen in Figure 4. The reader can find some core codes and main support of detection in Appendix A.

4.2 Vehicle Tracking Algorithm: Based on Deep_Sort

The aim of vehicle tracking is to track the movement of each detected vehicle (15, 16). The research algorithm is based on a state-of-the-art tracking algorithm, i.e., DeepSORT (17). The
tracking program matches detections in the previous and current frames and establishes the corresponding frame image. It is one of the most robust algorithms against occlusion through the use of appearance information. Deep_SORT is able to track objects through long periods of occlusion. Therefore it can effectively reduce the number of identity switches that would otherwise occur whenever images are blocked within a given video frame. For every tracked bounding box, we use the center coordinate to draw the trajectory. The trajectory is then used to calculate the pixel speed for speed estimation (see in Section 4.3). We show the output result of the tracking algorithm in Figure 5. The resultant videos are annotated with bounding boxes and moving trajectories. The reader can find some core codes and main support of tracking in Appendix A.

Figure 5. Output result of tracking algorithm.

4.3 Speed Estimation Algorithm

The aim of the speed estimation algorithm is to estimate the speed values of each tracked vehicle (18, 19, 20). The workflow of our algorithm is summarized as follows:

Workflow:

- Calibrate the first frame to get the rectification homography.
  - detect two vanishing points (horizontal & vertical)
  - calculate the homography matrix H
- Get the trajectory of each tracked vehicle.
- Transform the trajectory to rectification domain via H.
  - calculate the pixel speed (pixel/second)
- Assume every block on the highway has a fixed distance.
  - reference means the length of every pixel in real world (meter/pixel)
- Estimate the speed: reference \times pixel speed.

As used by our research, planar homography relates the transformation between two planes. Geometrically, an image captured with an arbitrarily rotated camera, can be thought of as a transform from the fronto-parallel image plane (seen in Figure 6). We use rectification homography, which makes the lines that were parallel in fronto-parallel view become parallel again in the rectification domain.

![Figure 6. Left: arbitrary view, Right: fronto-parallel view.](image)

Let $H$ denote the rectification homography. $H$ is a 3 by 3 matrix. Our rectification homography $H$ can transform the image back to the fronto-parallel plane by using matrix multiplication (results seen in Figure 7).
Next, for every tracked bounding box, we use the center coordinate to draw the trajectory. For each frame, we get the following matrices which contain the coordinates:

\[
[\text{vehicle ID}] [\text{position 1}] [\text{position 2}] [\text{position 3}] \ldots
\]

E.g., \([1] [x1,y1] [x2,y2] [x3,y3] \ldots\)

Here, \([\text{position 1}]\) denotes the coordinates of \#\(\text{vehicle ID}\) in the first tracked frame. \(\text{vehicle ID}\) indicates the tracked vehicle by assigning a number to it. Using lines to connect the coordinates, we can obtain the trajectory.

Since the values of estimated speed should be calculated in the rectification domain, transform is needed in our implementation. By using rectification homography \(H\), we transform the trajectories into the rectification domain by matrix multiplication. We get the following matrices which contain the coordinates in the rectification domain:

\[
[\text{vehicle ID}] [\text{position 1'}] [\text{position 2'}] [\text{position 3'}] \ldots
\]

E.g, \([1] [x1',y1'] [x2',y2'] [x3',y3'] \ldots\)

Here, \([\text{position 1'}]\) denotes the coordinates of \#\(\text{vehicle ID}\) in the first tracked frame after rectification.

The pixel speed is calculated in the rectification domain as:

\[
S_{\text{ps}} = F \times \sqrt{(x_2' - x_1')^2 + (y_2' - y_1')^2}
\]  \[1\]

where:

\(S_{\text{ps}} = \) pixel speed (pixel/second); and

\(F = \) frame rate (1/second).

According to the reference, we can transform pixel speed into speed in meters per second for real world speed detection.

\[
S = R \times S_{\text{ps}}
\]  \[2\]

where:
\[ S = \text{speed in real world (meter/second)}; \text{ and} \]
\[ R = \text{reference (meter/pixel)}. \]

After obtaining the speed values, the speed values are displayed directly on the image frame. So the observers can easily acquire these information. It is worth mentioning that 10 frames are needed to do the average and make the speed value stable. Researchers found that as vehicles passed each other, obscuration errors can occur due to missing frames and this is discussed in section 5.3. The reader can find some core codes of speed estimation in Appendix A.
5. ANALYSIS AND FINDINGS

5.1 Output Results
We used the UAV to capture some videos near interstate highway 10. Some captured videos are shared on google drive as shown in Figure 8 and Figure 9.

Demo1:
https://drive.google.com/file/d/1u8QrPP7yk4AJ9lwAH1ZR7olpGXqMTZcX/view?usp=sharing

Demo2:
https://drive.google.com/file/d/1VUjyDy17VAhQh8YNXwrQT6lDtgk4RHzK/view?usp=sharing

After several frames our software rectifies a bounding box and vehicle speed is detected and displayed. When the vehicle just enter the view (or leave the view), the speed value is not accurate enough due to the frequent change of bounding box. So the software doesn’t calculate the speed values during the first 10 frames of the video.

As illustrated in Figure 8, we show the output results of selected frames after running algorithms. Image frames with sparse vehicles are captured in the video shown in Figure 8. The vehicles are annotated with bounding boxes, moving trajectories and the speed values. The white bounding box is the detection box, and the bounding box with random color is the tracking box. For every tracked vehicle, the software draw a moving trajectory with the same color of tracking box. The speed values are displayed just above the bounding box with green color. As illustrated in Figure 9, this video captured some image frames with more vehicles. The software works well in this situation.

Figure 8. Output result of our algorithms (video1). The resultant videos are annotated with bounding boxes, moving trajectories and speed values.
5.2 Speed Estimation Experimental Results

For this part of our research, we verified the accuracy of our speed estimation algorithm. Some experimental settings are described as below:

- We captured new videos (Dr. Hassan drove the car with fixed speed passing through, and Dr. Malveaux flew the drone to capture videos).
- In total, we got 10 video clips: 3 at speed 25mph, 3 at speed 30 mph, 2 at speed 40 mph, and 2 at speed 50 mph.
- Ran the algorithms on the captured video clips, and compared the estimated speed values with the known values.

As illustrated in Figure 10 and Figure 11, the speed values are estimated by our speed estimation algorithm. The speed values are labelled with the text ‘speed: [XX] mph’ in green color. As we can see, when the real speed is 25 mph, our estimated speed values are 24.3 mph and 23.9 mph; when the real speed is 30 mph, our estimated speed values are 28.2 mph and 27.9 mph; when the real speed is 40 mph, our estimated speed values are 37.2 mph and 37.7 mph; when the real speed is 50 mph, our estimated speed values are 48.0 mph and 47.7 mph. In all situations, the speed values estimated by the software are slightly lower than the speed values displayed on the speedometer. According to the law, a speedometer must never read less than the actual speed or show more than 110% of actual speed. In other words, if the speedometer reads 50 mph, the actual speed value must less than 50 mph. It’s possible the vehicle might only travelling at 47 or 48 mph.
Figure 10. Speed estimation experimental results when the real speed values are 25 mph and 30 mph. Top: 25 mph, Bottom: 30 mph.

Figure 11. Speed estimation experimental results when the real speed values are 40 mph and 50 mph. Top: 40 mph, Bottom: 50 mph.

The speed values are also visualized in Figure 12 and Figure 13. As we can see, when the real speed is 25 mph (here the researchers simply regard the speed values on speedometer as the real speed), the estimated speed values are very close to 25 mph and fluctuate between 23 and 25 mph. Similar situation can be observed when the real speed is 50 mph. Accordingly, our estimated speed values are accurate.
5.3 Occlusion

Occlusion is very common in traffic tracking scenes. Our research found a failure case in our demo video (seen in Figure 14). In this case as a pickup truck overtakes the small car seen in the figure our speed estimation algorithm loses track of the car and can only track the larger vehicle. We plan to replace YOLOv3 with YOLOv4 for better detection accuracy.
Figure 14. When our algorithms meet occlusion.

5.4 Manual Annotation
Since the drone has only one camera, we have to manually input some values to the codes. In our implementation, we utilize the first frame of the video as the reference frame and label two things:

1. The reference: we need to know the real length of the red segment on the reference frame (see in the left image of Figure 15).

2. Vanishing points: we need to label four points which are the corners of a rectangle in real world (see in the right image of Figure 15).

Figure 15. Manual Annotation on the reference frame.

5.5 Traffic Management App
We plan to integrate the data collected by the UAV (e.g., average speed of vehicles) with a mobile app. We have conducted a literature review on existing mobile apps. Two such apps are listed in Figure 16 and Figure 17.
INRIX TRAFFIC is popular, and lets the users input information (21).

Pros:
- Keeps track of your routes and driving habits and calculates accordingly.
- Tells you the best time to leave and what route to take for frequently used drives based on traffic and road conditions.
- Users can report incidents like accidents and other road hazards.
- Gives alerts for traffic conditions, lane closures, closed roads, and accidents.
- Can also search for parking options.

Cons:
- Reports of road closures, construction or traffic jams not always up to date.
- Rerouting is not always quick.
- Users reported traffic camera option has been removed in some cities.
- Traffic information may be less accurate in less-populous places.
This well-know app, Google Maps, has live traffic information built into its navigation function (22).

Pros:
- Offers real-time traffic and navigation information.
- Gives ETAs and traffic conditions.
- Offers automatic rerouting due to traffic, accidents, or other road conditions.

Cons:
- Users have reported bugs that cause the app to crash or shut off without warning.
- Sometimes the automatic reroute option doesn’t work.
6. CONCLUSIONS

This project has proven that UAS monitoring of highway construction zones and real-time alerts to motorists, construction crews, and first responders is possible in concept. Calibration experiments demonstrated the ability to detect vehicle speed from video to within 2 mph of actual vehicle speed. A combination of commercially available machine learning algorithms, Matlab data processing software, and custom coding by researchers was successful in creating a method for processing and analyzing highway traffic flow data, and research generated custom software was successful in detecting vehicle speeds from zero mph to highway speeds.

Reviews of available mobile traffic apps were conducted and 2 applications, INRIX and Google Maps were chosen for future integration with advanced iterations of the UAV and software system that has been created by this research. Mobile applications interface will become a key way in which the traffic monitoring UAS will link with drivers, first responders, and construction crews to distribute information.

Future research is needed to take UAS traffic monitoring from the proof of concept developed for this research to a final product. Future work consists of automated determination of vanishing point coordinates for vehicle speed estimation, and integration with cellular applications along with automated data transfer to create real time traffic alerts. This research will explore the use of UAV’s to manage traffic through a cloud based internet of things style interface with cellular mapping applications. Utilizing commercially available UAV systems this future research will encourage the participation of local industry partners who have ready access to commercial off the shelf UAV products and thereby stimulate the economy while meeting the goals and objectives of TranSet.
REFERENCES


APPENDIX A: Codes and Configurations

- Code 1

We show the core codes of the detection algorithm as follows:

Core codes of detection:

```python
while True:
    ret, frame = video_capture.read()  # Frame shape 640*480*3
    if ret != True:
        break
    t0 = time.time()
    image = Image.fromarray(frame[...,::-1])  # BGR to RGB
    boxes, class_names = yolo.detect_image(image)  # Detection part, it returns the bbox and the category
    features = encoder(frame, boxes)  # This feature is used for tracking
    detections = [Detection(bbox, 1.0, feature) for bbox, feature in zip(boxes, features)]
    # Run non-maximum suppression.
    boxes = np.array([d.tlwh for d in detections])
    scores = np.array([d.score for d in detections])
    indices = preprocessing.non_max_suppression(boxes, nms_max_overlap, scores)
    detections = [detections[i] for i in indices]

Main Support:

class YOLO(object):
    def __init__(self):
        self.model_path = './model_data/yolo.h5'
        self.anchors_path = 'model_data/yolo_anchors.txt'
        self.class_path = 'model_data/coco_classes.txt'
    def detect_image(self, image):
```

- Code 2

We show the core codes of the tracking algorithm as follows:

Core codes of tracking:

```python
# Definition of the parameters
max_cosine_distance = 0.5
nn_budget = None
nms_max_overlap = 0.3

counter = []

# deep sort
model_filename = 'model_data/market1501.pb'
encoder = gdet.create_box_encoder(model_filename, batch_size=1)
# Generate the feature vector of each detection bounding box by a pretrained appearance model
metric = nn_matching.NearestNeighborDistanceMetric('cosine', max_cosine_distance, nn_budget)
tracker = Tracker(metric) # construct class Tracker - tracker

Main Support:
```
class Tracker:
    ...
    Parameters
    -----------
    metric : nn_matching.NearestNeighborDistanceMetric
        A distance metric for measurement-to-track association.
    max_age : int
        Maximum number of missed passes before a track is deleted.
    n_init : int
        Number of consecutive detections before the track is confirmed. The
        track state is set to 'deleted' if a miss occurs within the first
        `n_init` frames.

    Attributes
    ----------
    metric : nn_matching.NearestNeighborDistanceMetric
        The distance metric used for measurement to track association.
    max_age : int
        Maximum number of missed passes before a track is deleted.
    n_init : int
        Number of frames that a track remains in initialization phase.
    kf : cv2.KalmanFilter
        A Kalman filter to filter target trajectories in image space.
    tracks : List[Track]
        The list of active tracks at the current time step.
    ...

def __init__(self, metric, max_age=30, max_n_init=3):

- Code 3

We also show the core codes of speed estimation algorithm as follows:

Core Codes:

```python
def calculate_speed(p1, p2):
    ...
    :param p1: first position of tracked vehicle
    :param p2: second position of tracked vehicle
    :return: speed in real world

def compute_homography_and_warp(image, vp1, vp2, clip=True, clip_factor=3):
    """Compute homography from vanishing points and warp the image."
    It is assumed that vp1 and vp2 correspond to horizontal and vertical
directions, although the order is not assumed. First, projective
transform is computed to make the vanishing points go
to infinity so that we have a fronto parallel view. Then, Computes
affine transform to make axes corresponding to vanishing points orthogonal.
Finally, Image is translated so that the image is not missed. Note that
this Image can be very large. `clip` is provided to deal with this.

    Parameters
    ----------
    image: ndarray
        Image which has to be wrapped.
    vp1: ndarray of shape (3,)
        First vanishing point in homogenous coordinate system.
    vp2: ndarray of shape (3,)
        Second vanishing point in homogenous coordinate system.
    clip: bool, optional
        If True, image is clipped to clip_factor.
    clip_factor: float, optional
        Proportion of image in multiples of Image size to be retained if gone
        out of bounds after homography.
    Returns
    -------
    warped_img: ndarray
        Image warped using homography as described above.
```
Basic configuration of codes

Our codes are based on Python and Matlab.

The basic environment: Ubuntu18.04 + CUDA10.0 + Cudnn7.4 + Python3.5 + Nvidia RTX 2080Ti

Deep learning platform: tensorflow-gpu 1.13.0

Third party packages: Opencv-Python + Keras + Scipy + Numpy + Scikit-learn + Pillow + PIL