COMPUTER VISION FOR TRAFFIC MONITORING

FINAL PROJECT REPORT

by

Matthew Thompson, Michael Lowry, Ahmed Abdel-Rahim
University of Idaho

Sponsorship
University of Idaho

for

Pacific Northwest Transportation Consortium (PacTrans)
USDOT University Transportation Center for Federal Region 10
University of Washington
More Hall 112, Box 352700
Seattle, WA 98195-2700

In cooperation with U.S. Department of Transportation,
Office of the Assistant Secretary for Research and Technology (OST-R)

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 under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The Pacific Northwest Transportation Consortium, the U.S. Government and matching sponsor assume no liability for the contents or use thereof.

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
	01872763	

4. Title and Subtitle	5. Report Date
Computer Vision for Traffic Monitoring	May 2024
	6. Performing Organization Code

7. Author(s) and Affiliations	8. Performing Organization Report No.
Matthew Thompson, University of Idaho Michael Lowry, 0000-0001-8485-4799; University of Idaho Ahmed Abdel-Rahim, 0000-0001-9756-554X; University of Idaho	2022-S-UI-3

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
PacTrans Pacific Northwest Transportation Consortium University Transportation Center for Federal Region 10 University of Washington More Hall 112 Seattle, WA 98195-2700	11. Contract or Grant No. 69A3551747110

12. Sponsoring Organization Name and Address	13. Type of Report and Period Covered
United States Department of Transportation Research and Innovative Technology Administration 1200 New Jersey Avenue, SE Washington, DC 20590	08/15/2023 – 05/31/2024 14. Sponsoring Agency Code

15. Supplementary Notes

Report uploaded to: www.pactrans.org

16. Abstract

This project examined computer vision applications for traffic monitoring and safety analysis. The focus was on evaluating the open-source computer vision code that we developed. Three case studies were completed using our computer vision code. The code was written in Python and uses a detection model called YOLOv8. The first case study demonstrated how user counts can be obtained from video feeds and provided examples of insights that can be drawn from these counts. The second case study used computer vision to create visualizations of user movements at intersections. The third case study developed and demonstrated the application of a new surrogate safety measure for pedestrian and bicyclist safety. Shortcomings and future opportunities of open-source computer vision systems are discussed.

17. Key Words	18. Distribution Statement
computer vision, traffic monitoring, artificial intelligence, machine learning, pedestrian flow, cyclists	

19. Security Classification (of this report)	20. Security Classification (of this page)	21. No. of Pages	22. Price
Unclassified.	Unclassified.	31	N/A

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

SI* (MODERN METRIC) CONVERSION FACTORS

APPROXIMATE CONVERSIONS TO SI UNITS				
Symbol	**When You Know**	**Multiply By**	**To Find**	**Symbol**
		LENGTH		
in	inches	25.4	millimeters	mm
ft	feet	0.305	meters	m
yd	yards	0.914	meters	m
mi	miles	1.61	kilometers	km
		AREA		
in^2	square inches	645.2	square millimeters	mm^2
ft^2	square feet	0.093	square meters	m^2
yd^2	square yard	0.836	square meters	m^2
ac	acres	0.405	hectares	ha
mi^2	square miles	2.59	square kilometers	km^2
		VOLUME		
fl oz	fluid ounces	29.57	milliliters	mL
gal	gallons	3.785	liters	L
ft^3	cubic feet	0.028	cubic meters	m^3
yd^3	cubic yards	0.765	cubic meters	m^3
	NOTE: volumes greater than 1000 L shall be shown in m^3			
		MASS		
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2000 lb)	0.907	megagrams (or "metric ton")	Mg (or "t")
		TEMPERATURE (exact degrees)		
°F	Fahrenheit	5 (F-32)/9 or (F-32)/1.8	Celsius	°C
		ILLUMINATION		
fc	foot-candles	10.76	lux	lx
fl	foot-Lamberts	3.426	$candela/m^2$	cd/m^2
		FORCE and PRESSURE or STRESS		
lbf	poundforce	4.45	newtons	N
lbf/in^2	poundforce per square inch	6.89	kilopascals	kPa

APPROXIMATE CONVERSIONS FROM SI UNITS				
Symbol	**When You Know**	**Multiply By**	**To Find**	**Symbol**
		LENGTH		
mm	millimeters	0.039	inches	in
m	meters	3.28	feet	ft
m	meters	1.09	yards	yd
km	kilometers	0.621	miles	mi
		AREA		
mm^2	square millimeters	0.0016	square inches	in^2
m^2	square meters	10.764	square feet	ft^2
m^2	square meters	1.195	square yards	yd^2
ha	hectares	2.47	acres	ac
km^2	square kilometers	0.386	square miles	mi^2
		VOLUME		
mL	milliliters	0.034	fluid ounces	fl oz
L	liters	0.264	gallons	gal
m^3	cubic meters	35.314	cubic feet	ft^3
m^3	cubic meters	1.307	cubic yards	yd^3
		MASS		
g	grams	0.035	ounces	oz
kg	kilograms	2.202	pounds	lb
Mg (or "t")	megagrams (or "metric ton")	1.103	short tons (2000 lb)	T
		TEMPERATURE (exact degrees)		
°C	Celsius	1.8C+32	Fahrenheit	°F
		ILLUMINATION		
lx	lux	0.0929	foot-candles	fc
cd/m^2	$candela/m^2$	0.2919	foot-Lamberts	fl
		FORCE and PRESSURE or STRESS		
N	newtons	0.225	poundforce	lbf
kPa	kilopascals	0.145	poundforce per square inch	lbf/in^2

*SI is the symbol for the International System of Units. Appropriate rounding should be made to comply with Section 4 of ASTM E380. (Revised March 2003)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

This project examined computer vision applications for traffic monitoring and safety analysis. The focus was on evaluating the open-source computer vision code that we developed. Three case studies were completed using our computer vision code. The code was written in Python and uses a detection model called YOLOv8. The first case study demonstrated how user counts can be obtained from video feeds and provided examples of insights that can be drawn from these counts. The second case study used computer vision to create visualizations of user movements at intersections. The third case study developed and demonstrated the application of a new surrogate safety measure for pedestrian and bicyclist safety. Shortcomings and future opportunities of open-source computer vision systems are discussed.

CHAPTER 1. INTRODUCTION

1.1. Problem Statement

In the 1970s, academics began exploring the possibility of analyzing image pixels to detect vehicles in video footage. By the early 2000s, advancements in image processing facilitated the widespread adoption of video detection systems in traffic management throughout the United States. These early systems involved calibrating the background image against the changing pixels inside specific zones on the video image to determine vehicle counts and turning movements. These early systems focused on detecting automobiles and trucks, while bicycles and pedestrians were ignored outside academic work. Another twenty years passed, and the field of computer vision emerged. Computer vision (CV) is a branch of artificial intelligence (AI) that uses machine learning and neural networks to help computers detect and track objects in visual data. Computer vision provides exciting traffic monitoring and safety analysis opportunities, especially for pedestrians and bicyclists.

Various companies offer computer vision analytics for traffic monitoring and safety, but their tools and services can be too costly for some municipalities. Recently, free, open-source computer vision tools have emerged, which might provide a cost-effective way for some cities to use computer vision to diagnose intersections that threaten pedestrian and bicyclist safety. Many communities rely on federal grant funding to improve their transportation infrastructure, making it essential to apply as much of this funding as possible to their projects rather than spending it on identifying or justifying projects. Engineers need quicker and more accessible tools, especially for safety concerns regarding vulnerable roadway users such as bicyclists and pedestrians.

1.2. Objectives

The objectives of this research were as follows:

1. Develop open-source computer vision tools.
2. Identify shortcomings and advantages of open-source computer vision.

1.3. Organization

Chapter 2 provides background on traffic monitoring and safety analysis for bicycles and pedestrians. Chapter 3 presents the custom CV analysis techniques done in Python. Chapter 4 describes conclusions that can be made from this study and suggestions for further research in applying computer vision technology to monitoring the safety of vulnerable roadway users.

Lastly, Appendix A provides more information on the research that was done leading up to the development of this analysis system.

1.4. Contribution of the Research

This project explored computer vision applications for the transportation engineering community. Additionally, this project evaluated the benefits and shortcomings of computer vision, particularly for pedestrian and bicyclist studies. Lastly, this research introduced a new metric for evaluating the degree to which vulnerable roadway users are exposed to risky situations.

CHAPTER 2. LITERATURE REVIEW

This chapter describes traditional traffic monitoring and safety analysis methods and recent studies that have explored the potential for computer vision.

2.1. Traffic Monitoring

Traffic monitoring collects information such as vehicle volumes, classification, and speed (FHWA, 2022). Agencies want this information in hourly or daily increments to make decisions to improve and manage traffic (Fontaine et al., 2009). Figure 2.1 compares various traffic monitoring technologies.

Category	Sensing type	Cost	Energy Consumption	Easy to Install	Traffic flow Parameters
Intrusive sensors	Pneumatic Tube	Low	High	No	Count, Classification
	Inductive Loop	Low	High	No	Count, Speed, Classification
	Piezoelectric sensors	Low	High	No	Count, Speed, Classification
	Magnetic sensors	Low	Low	Yes	Count, Speed, Classification
Non-Intrusive sensors	Accelerometers	Medium	Low	Yes	Count, Speed
	Acoustic sensors	High	Low	Yes	Count, Speed, Classification
	Infrared sensors	High	Low	Yes	Count, Classification
	Radar	High	High	Yes	Count, Speed, Classification
	Ultrasonic sensors	High	Medium	Yes	Count, Speed
	Wi-Fi	Low	Low	Yes	Count, Classification
	Bluetooth	High	Low	Yes	Count, Classification

Figure 2.1 Traditional traffic monitoring techniques.
Source: Khattak et al. 2022. (Figure licensed under CC BY 4.0)

Sensing technologies are either "intrusive" or "non-intrusive." Intrusive technologies are installed directly onto or within the pavement surface. Non-intrusive technologies are set up on the roadside or in an overhead position. In the 1990s, a popular non-intrusive technology called video detection emerged. Cameras are positioned at an intersection, and the system is calibrated to match the background image. Vehicles are detected on the basis of changing pixels inside specific zones. These early systems focused on detecting automobiles and trucks (Bullock et al., 2002). There has been considerable bias toward automobiles in traffic monitoring, primarily because of technological issues. Lee and Sener (2020) acknowledged that non-motorized travel

modes have unique characteristics, such as higher sensitivity to the environment (e.g., weather, topography), higher variation in route choice, and shorter durations of trips.

Computer vision is an advanced video detection method that uses artificial intelligence (AI). This new technology is an improvement in three ways:

- It does not require calibrating to the background image.
- Objects can be detected anywhere in the field of view.
- Pedestrians and bicyclists can be detected with high accuracy.

The applicability of computer vision to transportation engineering is still in the process of being discovered. One application is for real-time incident response. Another application is for autonomous vehicles (Dilek et al., 2023).

Several studies have employed computer vision for pedestrian traffic monitoring. Pourhomayoun (2020) found an average hourly count error of 4.1 percent for 12 hours in comparison to ground truth. Wong et al. (2021) analyzed pedestrian speed and crowd density for implementation in pedestrian simulations and to understand more about walking spaces. Karagulian et al. (2023) sought to understand pedestrian characteristics by analyzing pedestrian speed, direction, and density within a pedestrian plaza in Italy. As Molina et al. (2022) indicated, it is beneficial to visualize object movements to better understand roadway congestion. Such visualization cannot be obtained through other monitoring techniques, such as pneumatic tube counters.

Research has also been done regarding bicycle detection with computer vision. Rogers and Papanikolopoulos (2000) sought to obtain information about the usage and congestion of bicycle paths with a rudimentary system that used gray-scale images and was trained to look for two circles separated by a range of distances. This system was viable because the camera was mounted perpendicularly to a bike path, meaning that each image was a profile, and the system was intended only to obtain count metrics, which was done with around 70 percent accuracy. Another such system put forth by Takahashi et al. (2010) looked at identifying bicycles on the basis of the leg movements that occur during the pedaling motion. This decision was made to more accurately distinguish bicycles from motorcycles and people, with an accuracy of 76 percent. Moro et al. (2011) sought to develop a model capable of distinguishing cars, pedestrians, and bicycles. The system was tested for its performance in detecting bicycles, and true positives were detected 78.5 percent of the time and false positives 3.2 percent of the time. Li et al. (2014)

sought to quantify the extent to which bicyclists were wearing helmets while riding. Their study required the ability to detect bicyclists accurately and then identify whether or not the bicyclist was wearing a helmet. The first step of this process was using a model trained on cyclists' oscillatory movement patterns in comparison to primarily linear vehicle movements. Helmet presence was next detected on the basis of the color, shape, and texture of a region identified as the head region of the bicyclist.

2.2. Safety Analysis

Pedestrian and bicyclist safety remains a topic of high importance for engineers, policymakers, and the public. Data provided by the National Highway Traffic Safety Administration (NHTSA) revealed that in 2021, 7,388 pedestrians were killed in traffic crashes, and another 60,577 pedestrians were injured in the same year, representing a 12.5 percent and 11 percent increase from the previous year, respectively (National Center for Statistics and Analysis, 2023b). Bicyclist data for the same year indicated that 966 bicyclist fatalities and 41,615 bicyclist injuries occurred. (National Center for Statistics and Analysis, 2023a). There has been considerable bias toward automobiles in safety analysis, mainly because of technological issues.

There are notable differences between vehicle-vehicle conflicts and those that involve a pedestrian. Cynecki (1980) highlighted 11 such differences, as shown in Figure 2.2.

1. Vehicle-pedestrian accidents are not as numerous as vehicle-vehicle or vehicle-fixed object accidents.
2. Accidents are considerably more severe when they involve pedestrians.
3. Many of the pedestrians involved in accidents are school age, preschool or elderly.
4. The operation of a motor vehicle is considerably more complex and involves many more distractions than does the action of a pedestrian crossing a roadway.
5. Pedestrians are more maneuverable than automobiles in the actions of stopping and changing directions.
6. The speed differential between automobiles and pedestrians is considerable.
7. Most pedestrian conflicts occur at right angles when pedestrians are in the action of crossing a roadway. This is not the case with vehicle conflicts.
8. Pedestrians may access a roadway at almost any point.
9. Sight restrictions may be more prevalent in pedestrian conflicts than in vehicle conflicts.
10. For nighttime conditions, vehicles have headlights, taillights, and running lights that make them more visible to other motorists. Pedestrians are often dressed in dark clothing and are more difficult to detect at night.
11. Pedestrian conflicts and conflict severity can be measured from the movements and reactions of vehicles and pedestrians. When only vehicles are involved, conflicts and conflict severity can usually be measured by watching the movements and reactions of only one vehicle.

Figure 2.2 Pedestrian safety analysis special needs.
Source: Adapted from Cynecki, 1980

The conventional method of safety analysis uses police reports of crashes. Criticisms of this method of looking at crash statistics include, "Factual discrepancies, delayed data collection, and different standards for constructing a report" (Jones, 2023). For a long time, there has been a basic understanding that dangerous interactions can occur without a crash. This concept has been applied previously in fields such as industrial safety and air travel. Klebelsberg (1964) observed an inverse relationship between the riskiness of traffic events and the frequency with which they occurred. Standard driving behavior occurred frequently and was not risky, whereas accidents occurred infrequently but were highly risky.

Hydén (1987) took this concept a step further and constructed a pyramid to show the relationship between the frequencies of these events. Hydén proposed, "There are elementary events, defined as serious conflicts, that can be characterized as breakdowns in the interaction [between road users]. The accident potential is then well-defined, i.e., there exists a relationship between the number of serious conflicts and accidents." Hydén also highlighted the need for

other traffic safety measures by saying that "in many applications, the accident numbers are so small that they easily lead to misinterpretations. In order to increase the numbers, one has to increase the number of applications or increase the analysis period" (Hydén, 1987). Hydén further illustrated this point with a pyramid representing the frequency of events. Fatal accidents made up the top of this pyramid, and the events decreased riskily until finally, potential conflicts and undisturbed passages made up the bottom of the pyramid. Pedestrians and bicyclists are represented by a scaled-down version of this pyramid, meaning that analysis of reported crashes is insufficient and surrogate measures are necessary.

Allen et al. (1978) described some early methods used in traffic conflict techniques to quantify dangerous events. They provided six methods: Proportion of Stopping Distance (PSD), Gap Time (GT), Encroachment Time (ET), Deceleration Rate (DR), Post Encroachment Time (PET), and Initially Attempted Post Encroachment (IAPE). These methods largely relate to vehicle/vehicle conflicts but have formed the basis for a field of study that analyzes dangerous encounters that do not result in crashes. Descriptions of each of these metrics are shown in Table 2.1.

Table 2.1 Surrogate safety measures

Measure of Safety	Description
PSD	The ratio of the distance available for a driver to maneuver to the distance remaining to the projected location of the collision
GT	T3-T2: where T3 denotes the time the through vehicle would be expected to arrive at the potential collision point by maintaining speed and direction. T2 is the time at which encroachment by the left-turning vehicle on the through lane ends
ET	The time during which the left-turning vehicle infringes upon the right-of-way of the through vehicle.
DR	Rapid deceleration indicates a severe conflict, whereas moderate deceleration normally implies a minor occurrence.
PET	The time from the end of encroachment to the time that the through vehicle arrives at the potential point of collision.
IAPE	Drivers commonly accelerate during the termination of a conflict. This measure uses the initial deceleration rate to forecast the initially attempted post-encroachment time.

Gettman and Head (2003) offered additional measures of conflicting traffic and analyzed their presence through traffic simulation models. The authors recommended collecting five surrogate measures for each traffic conflict: Time to Collision (TTC), Post Encroachment Time (PET), Initial Deceleration Rate, and the maximum speeds and relative speeds of the two vehicles involved in the conflict. Once again, these surrogate measures focus primarily on conflicts that occur between two vehicles rather than those involving pedestrians or bicyclists, but they further advance the methods available to engineers seeking to quantify safety. Gettman and Head (2003) also acknowledged other surrogate safety measures that "have not been quantitatively linked to crash rates but, rather, assert rules of thumb, such as more stop-bar encroachments indicate a higher probability of crashes; longer queues indicate a higher probability of crashes, and so on." Those measures include delay, approach speed, queue length, stop bar encroachments, red violations, speed, and deceleration distribution.

Research has been done to apply computer vision to traffic safety analysis. Zaki et al. (2013) diagnosed pedestrian safety issues using the TTC metric to quantify areas of an intersection where conflicts occurred and produce a heatmap of those interactions. The technology also allowed them to identify rates of pedestrian jaywalking violations with an accuracy above 85 percent. They determined that the intersection would benefit from geometric modifications and increased speed limit enforcement and jaywalking enforcement.

Xu et al. (2018) sought to use existing traffic monitoring cameras to perform an automated pedestrian safety analysis. Cameras in the City of Austin were analyzed with YOLOv2 on a high-performance computing cluster. They developed a system in which data could be easily queried on the basis of period, camera location, and user type to identify jaywalking safety concerns. Sayed et al. (2013) analyzed vehicle-bicycle interactions using the TTC indicator at a location in British Columbia that was perceived to have a high rate of vehicle/bicycle conflicts. Objects were tracked, and a perspective transformation was applied, allowing tracks to be assigned speeds and the distances between objects to be measured. Conflicts were categorized by severity, showing how often incidents of various TTC were occurring. This information was also plotted as a heat map to demonstrate which areas of the intersection posed a threat to bicyclists. Smaldone et al. (2010) sought to enhance bicyclist safety by developing an enhanced bicycle with onboard sensors that used computer vision and audio recognition to detect rear-approaching automobiles and alert the bicyclist to a vehicle's presence.

The authors asserted that the most dangerous situation for a bicyclist is being passed by a car from behind, and this technology would allow them to maintain situational awareness. An essential component of this research was training the model to distinguish between approaching and departing vehicles.

CHAPTER 3. DEVELOPMENT OF OPEN-SOURCE COMPUTER VISION TOOLS

This chapter describes the open-source computer vision code we developed for traffic monitoring and safety analysis.

3.1. Description of YOLO Python Algorithm

Our computer vision code is based on the open-source object detection architecture called YOLOv8. The YOLO architecture created by Joseph Redmon at the University of Washington in 2016 stands for You Only Look Once. Contrary to previous object detection algorithms, the YOLO algorithms perform all detections with a single pass, which results in slightly decreased accuracy but vastly increases the speed with which analysis can be done. The YOLOv8 framework, which was used for our Python analysis system, was created by a company called Ultralytics in January 2023. Ultralytics released five sizes of the model. Each of these models can be trained with additional images to enhance performance. The model names and their relative complexity are represented in Figure 3.1.

Figure 3.1 Visual representation of YOLOv8 model size.

The base models range in computation time and accuracy, with the larger models being more accurate but taking much longer to perform detections. Figure 3.2 plots each model's accuracy on the vertical axis (measured by mean average precision) vs the time required for predictions on a given graphics processing unit (GPU) (measured in milliseconds per image).

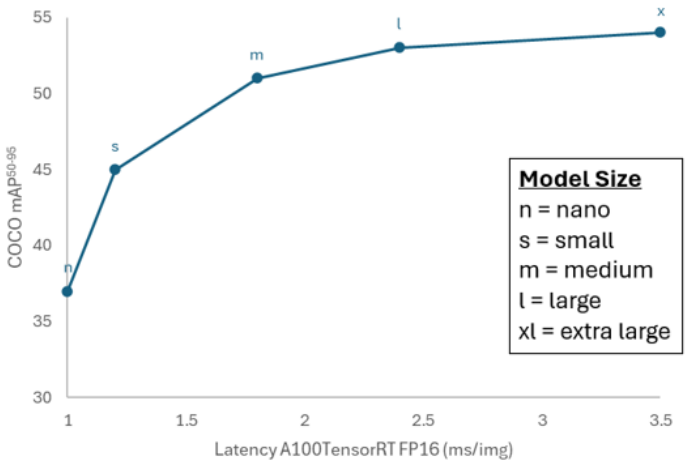


Figure 3.2 Comparison of different models.
Source: Adapted from Ultralytics (2023)

Each YOLO model was trained with the Common Objects in Context (COCO) dataset, consisting of over 300,000 images and containing 80 object categories. The object types relevant to traffic monitoring and their YOLO object class number are shown in Table 3.1.

Table 3.1 YOLO object classes

Object Class Name	YOLO Class Number
person	0
bicycle	1
car	2
motorcycle	3
airplane	4
bus	5
train	6
truck	7

When the analysis is run, the Python script identifies all objects for each video frame. It assigns these objects an object ID, an object class, bounding box coordinates, and a degree of confidence that the detection algorithm has in asserting that the object is of that type. The result is a table containing this information that can be used for multiple types of analysis. Figure 3.3 shows an example of the information provided by the detection algorithm. A confidence value

closer to 1 indicates that the detection model is more confident that the object is of the class that it states.

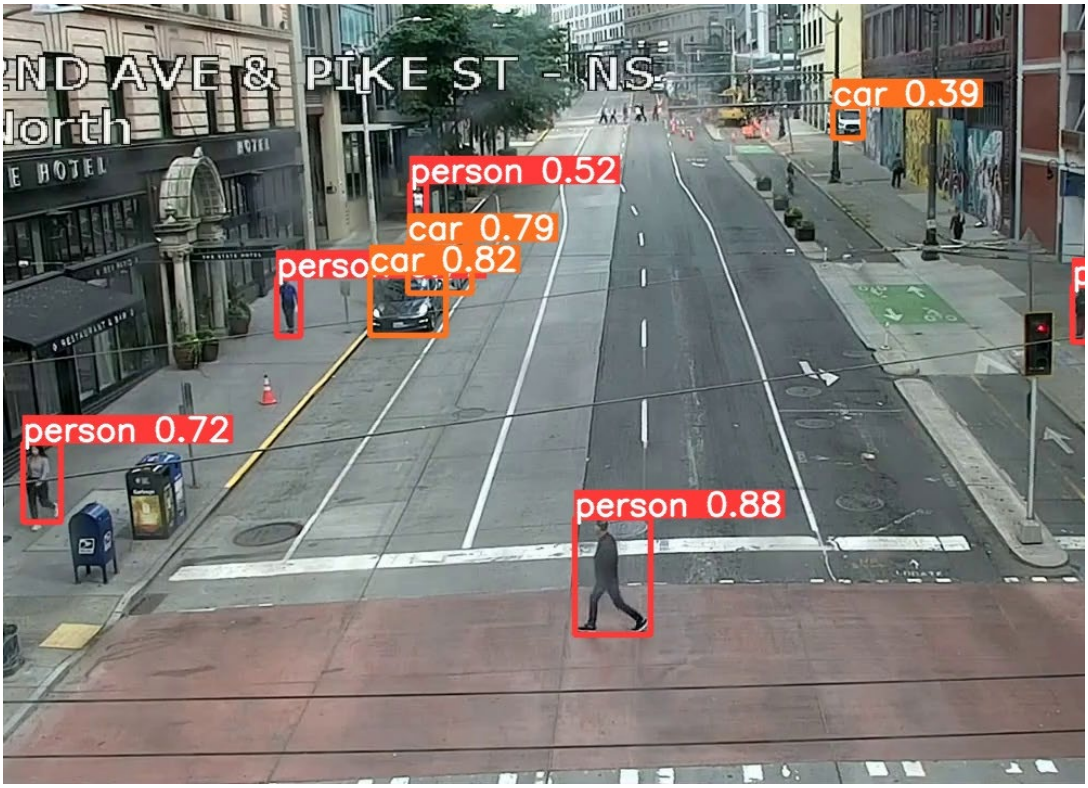


Figure 3.3 Example YOLO detection.

A compelling issue in computer vision is the idea of perspective. Depending on the height and angle of a camera, bounding box centroids may not be the ideal indicator of an object's location relative to a zone. As a result, the decision was made to identify an object's location by using the center of the bottom border of an object's bounding box.

3.2. Analysis Locations

Several video sources were used throughout this study to develop and test the detection and analysis system. Early in the process, a system was developed to extract videos from YouTube livestreams to analyze. This allowed for the observation of cameras in diverse areas and intersection types. For example, Figure 3.4 includes streams for a roundabout in California, a quiet market area in Florida, a busy Tokyo intersection, and an intersection of two highways in suburban Michigan. This is just a small subset of the types of places that can be viewed through analysis of YouTube livestreams. More about the process of extracting these feeds is included in Appendix A.

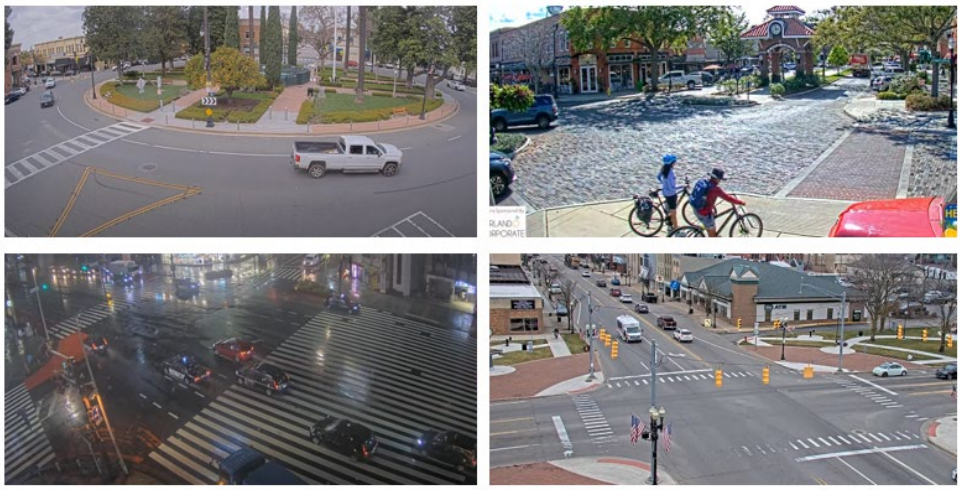


Figure 3.4 Example YouTube livestream locations.

Additionally, hundreds of cameras in Seattle were accessible through web scraping (see Appendix A). These cameras showed several busy urban intersections and helped test the system, as there was almost always a roadway user on camera to be detected. Figure 3.5 is a map of the downtown camera locations obtained from the Seattle Department of Transportation.

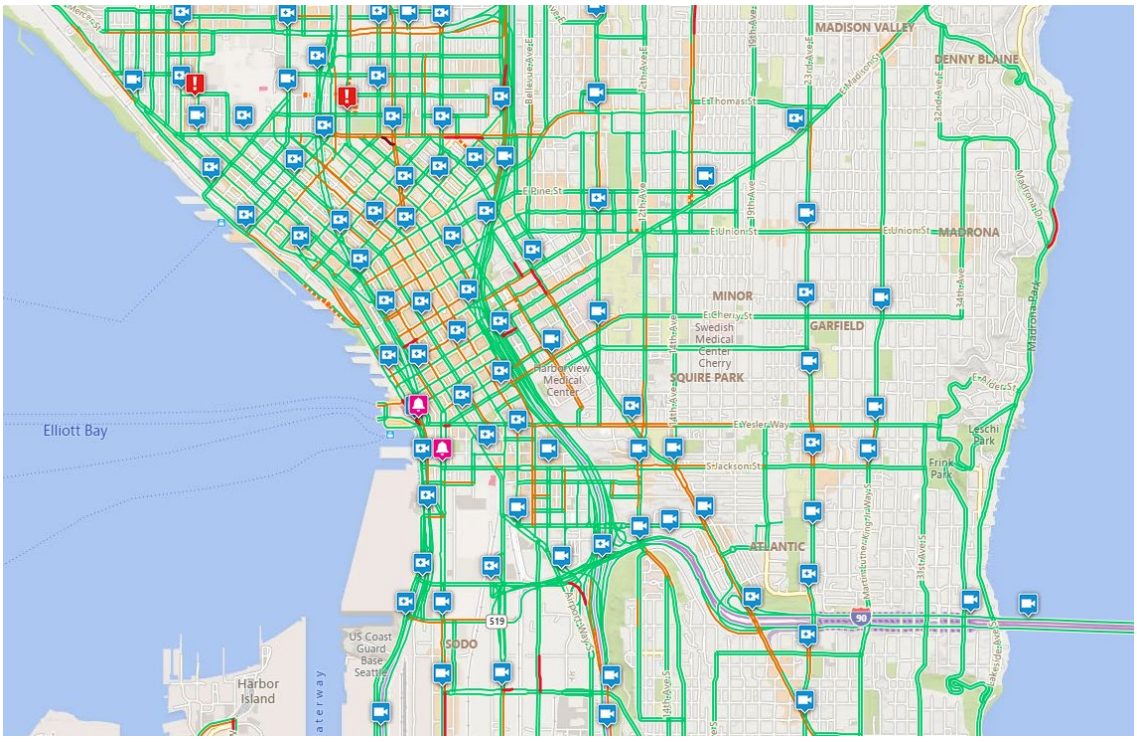


Figure 3.5 Example Seattle livestream locations.

Lastly, analysis was done in Moscow, Idaho, in cooperation with the Idaho Transportation Department (ITD). ITD has 274 cameras along its roadway network throughout Idaho. These cameras are used to monitor traffic incidents to coordinate response and to allow the public to check roadway conditions ahead of their travel. Most of these cameras are located in non-urban areas and are pointed at the interstates and highways that the state maintains. As such, many of these cameras are not monitoring roads that accommodate pedestrians and bicyclists.

The City of Moscow, Idaho, located in Latah County and home to the University of Idaho, contains two highways important for commerce throughout the state and the region. The first is SH-8, which runs from the Washington border to Elk River, Idaho (50 miles east of Moscow). The other major highway is US 95, which runs from I-80 in Winnemucca at its southern end to the Canadian border between Idaho and British Columbia at the northern end. Moscow, a small town of approximately 25,000 people, is unique in that it has high pedestrian and bicyclist mode shares while still being located along major regional roadways. This has forced the city and ITD to balance these demands to provide usable infrastructure for all. As a result, 13 cameras are located on roads within Moscow. Their locations are pictured in Figure 3.6, and they provide footage of facilities that pedestrians, bicyclists, and automobiles use. These previously described factors made Moscow an ideal site for analyzing the new techniques discussed in this report.

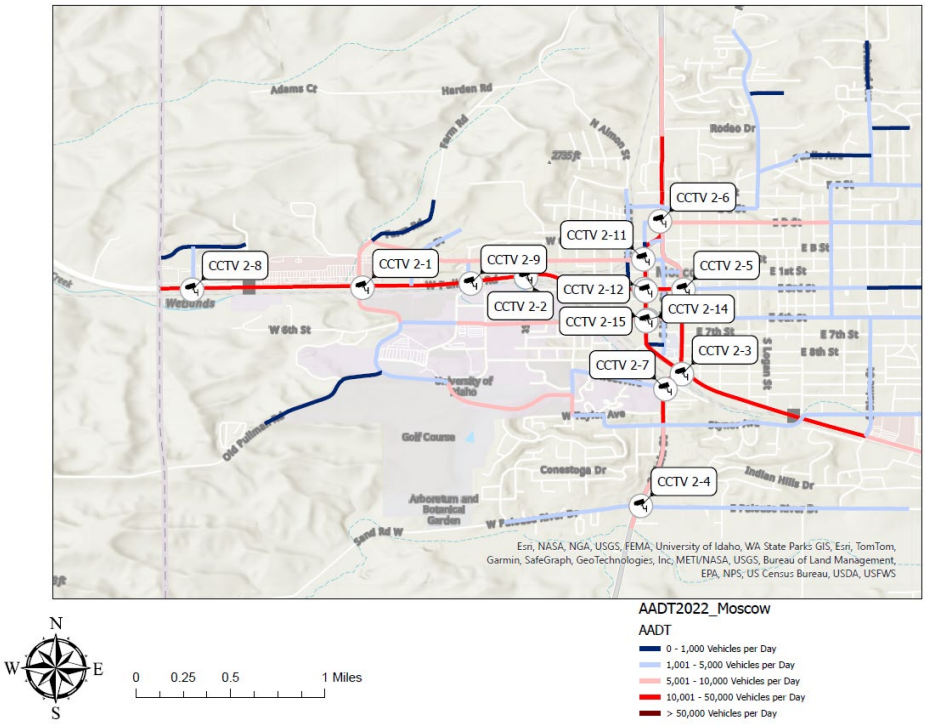


Figure 3.6 ITD cameras in Moscow.

3.3. Application 1: Traffic Monitoring

For this research, a series of functions were developed in Python to perform several tasks related to computer vision. The first of these tasks was to conduct a case study of traffic monitoring at locations in Moscow, Idaho. This analysis compared hourly volumes at different periods for two locations. These locations were referred to as Camera A and Camera B. Camera A was located in downtown Moscow at the intersection of 6th Street and Jackson Street. The ITD camera used for analysis at this location looked north up Jackson Street, a one-way road with three southbound through-lanes and a designated right turn lane (see Figure 3.7).

Figure 3.7 Jackson and 6th Street location.

The second location used in this case study was the intersection of Stadium Drive and SH-8 (the Moscow-Pullman Highway). The camera was located on the southeast corner of the intersection and was pointed in a northwesterly direction such that a view was provided of the southbound lanes of Stadium Drive and the westbound lanes of SH-8. South of the camera was the University of Idaho campus. North of it were several apartment buildings where students lived and commuted by foot to the University of Idaho campus (see Figure 3.8).

Figure 3.8 SH-8 and Stadium Drive location.

For each of these two cameras, zones of interest were created using an interactive Python tool we developed. This tool takes inputs of a base image for the camera, and the user clicks on the image four times in a clockwise or counterclockwise direction to create a closed polygon. These coordinates are then saved to a .txt file, and the polygon is plotted on the image. Two zones were created for Camera A, one of which covered the northern crosswalk of Jackson Street and the other covered the visible portion of the eastern crosswalk of 6th Street. The first of these zones could count the vehicles traveling on Jackson Street and pedestrians crossing Jackson Street on the north side of 6th Street. The second zone counted the vehicle traffic heading west on 6th Street and any pedestrians crossing 6th Street on the east side of Jackson Street.

For Camera B, three zones were drawn. The first of these zones was the crosswalk on the north side of the intersection. This zone counted southbound traffic and pedestrians crossing Stadium Drive on the north side of SH-8. The second zone covered the entire visible portion of the crosswalk on the west side of the intersection. This zone would get vehicle counts on SH-8 and pedestrian counts of those crossing SH-8 on the west side of the intersection. Lastly, a zone was drawn over just the westbound lanes of SH-8 to isolate the vehicle traffic moving in that direction.

Two hours of footage were analyzed for each location: one hour in the morning and one hour in the afternoon. Points were sorted by object, and each object was checked to identify whether it had points contained within the zone polygons. This allowed us to produce tables showing counts for each user type. Examples are shown in tables 3.2 and 3.3.

Table 3.2 Camera A counts

	Morning		**Afternoon**	
Object Class	**Zone 1**	**Zone 2**	**Zone 1**	**Zone 2**
Bicycle	9	6	4	1
Bus	37	0	13	1
Car	723	98	900	199
Motorcycle	0	3	20	0
Person	47	7	46	10
Truck	105	3	153	21
Train	6	0	0	0

Table 3.3 Camera B counts

	Morning			Afternoon		
Object Class	**Zone 1**	**Zone 2**	**Zone 3**	**Zone 1**	**Zone 2**	**Zone 3**
Bicycle	0	0	0	5	1	3
Bus	4	0	3	3	0	3
Car	845	147	521	1110	219	657
Motorcycle	0	0	0	4	1	3
Person	15	4	18	30	9	28
Truck	211	23	39	213	14	48

Several insights can be gathered from a count table like the ones above. For example, charts similar to those created by proprietary sensors can be created to visualize changes in usage for different times of day or to display the composition of road user types for a particular intersection. Given enough data, these charts can provide interesting insights about how road usage changes spatially or temporally. Engineers can use temporal insights to adjust signal timing parameters and spatial insights to emphasize infrastructural improvements in higher usage areas. Figure 3.9 compares the morning and afternoon at Camera A. The pedestrian counts were relatively similar, while car/truck counts increased in the afternoon. This could inform city engineers that a longer green phase might be necessary in the afternoon to accommodate higher volumes.

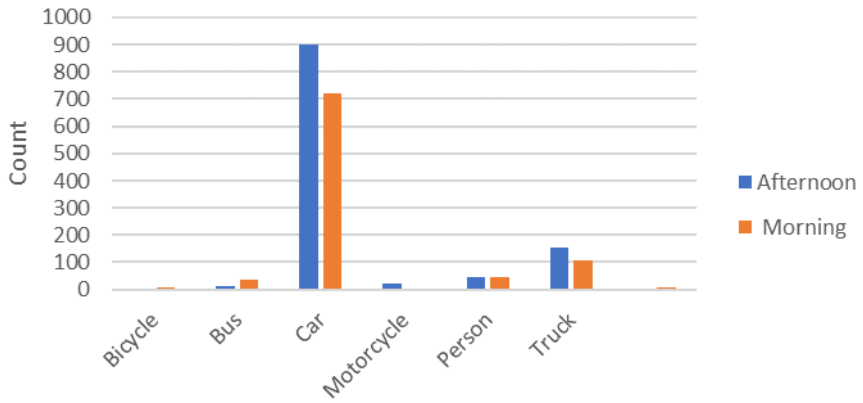


Figure 3.9 Comparing morning and afternoon counts for zone 1 of Camera A.

It is also important to note the current weaknesses of computer vision. Table 3.2 shows that the detection system allegedly saw six trains in an hour on Camera A. This can certainly be identified as an error in the system, as there are no train tracks near this intersection and it

mistakenly identified trucks or buses as trains. At this point computer vision is relatively susceptible to misidentifications and missed identifications. While we did not perform validation to conclude to what extent these misidentifications impacted user counts, we can certainly recognize that these shortcomings of the technology hinder our ability to confidently make decisions based on data obtained through computer vision. These errors can occur for a few different reasons:

- Occlusion (when an object is partially hidden by another object or the edge of the view).
- Objects may appear slightly different than they did in the training dataset (for example, many city buses look different, and the YOLO model may have an idea of what a bus looks like in NYC but not in Moscow, Idaho).
- Lighting or weather conditions.
- Objects may not fit into one of the defined object types (YOLO was not trained on electric scooters and might identify it as a person, a bicycle, or a motorcycle).

The good news related to these shortcomings is that computer scientists are constantly creating ways to reduce the effects of these circumstances and, thus the rate at which misidentifications and missed identifications occur. The error rate is certainly higher than it is for conventional methods such as tube counters or inductive loops; however, there is hope that computer vision can reach the same level of accuracy, accompanied by the additional insights not provided by those methods.

3.4. Application 2: Desire Line Heat Maps

A potential benefit of computer vision technology over traditional traffic monitoring methods is in visualizing traffic patterns. These visualizations give a sense of what paths roadway users are taking. High- and low-density areas can be seen by looking at the plotted paths of objects. Particularly with pedestrians and bicyclists, these visualizations can help detect behaviors that are outside intended use cases, such as pedestrians jaywalking or bicyclists riding on sidewalks (i.e., desire lines). When created for long durations, these heat maps can show general trends of usage while also allowing us to see deviations from those general trends and the frequency at which those deviations occur. The tool we created allowed us to choose which user types we wanted to display. For example, the heat map in Figure 3.10 shows the paths of all detected roadway users.

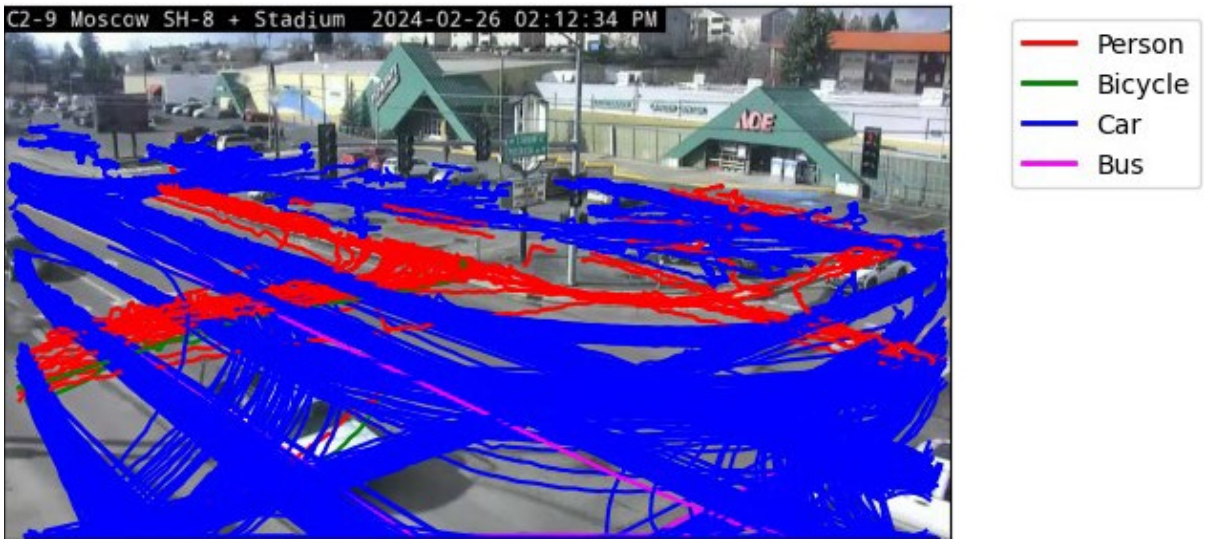


Figure 3.10 Desire line heat map for all user types.

Figure 3.11 is a heat map done for the same time period with isolated pedestrian and bicyclist movements. The ability to distinguish these user types and visualize them separately is helpful because automobile volumes are often orders of magnitude higher than pedestrian and bicyclist volumes, meaning that these vulnerable users get lost in the heat maps that show all user types.

Figure 3.11 Desire line heat map for pedestrians and bicyclists.

An important thing to note about computer vision is that it has an easier time detecting objects that are closer to the camera, and so there are some areas on the edges where it may not

have detections or where there are lapses in the detections of an object. This does not indicate that pedestrians or bicyclists avoid this area; the computer does not see them well enough to identify them definitively. Heat maps can visualize behaviors such as pedestrians angling through traffic lanes rather than completing their entire crossing in the crosswalk or bicyclists crossing in crosswalks versus traffic lanes. Engineers can use this information to make geometric or operational modifications to encourage more efficient and safer usage.

3.5. Application 3: Safety Analysis—New Surrogate Measure

One approach to safety analysis is to examine surrogate safety measures when there is a low frequency of crashes at a location, for example crashes that involve pedestrians and bicyclists. Rather than waiting for enough crashes to happen to constitute a viable sample for statistical analysis, engineers turn to surrogate measures of safety. Many of these measures require sophisticated systems that can interpret distances and speeds, and these data are quite difficult to obtain, particularly for pedestrians and bicyclists. In this case study, a new surrogate measure of exposure to dangerous situations for pedestrians and bicyclists was proposed called Conflicted Zone Exposure (CZE). The measure is defined by the following equation when looking at an individual bicycle or pedestrian *i,* where $f_{n,i}^{*}$ is a frame in which the traveler occupies a zone at the same time as a vehicle and $f_{n,i}$ is a frame in which the traveler is in the zone. The summation represents the total number of frames in which a vulnerable roadway user is in conflict.

Equation 3.1

$$CZE_i = \frac{\sum_n f_{n,i}^{*}}{\sum_n f_{n,i}} * 100$$

A vulnerable user is considered to be in conflict when there is a vehicle in the zone while the vulnerable user inhabits the zone. When looking at an intersection as a whole, the equation is as follows, where $\overline{CZE}$ is the zone-time weighted average. Each vulnerable user during the observation period is considered.

Equation 3.2

$$\overline{CZE} = \frac{\sum_i \sum_n f_{n,i}^{*}}{\sum_i \sum_n f_{n,i}} * 100$$

An example of a conflicted zone incident is in Figure 3.12, where a vehicle is turning left from Line Street onto SH-8 between two sets of crossing pedestrians. While this maneuver is not

necessarily dangerous, a situation might arise in which a driver only sees pedestrians crossing from one direction. We hypothesize that understanding these kinds of events can increase our understanding of pedestrian and bicyclist crashes without waiting for them to occur. Another example of what a conflicted zone might look like is a stop bar encroachment, as shown in Figure 3.13.

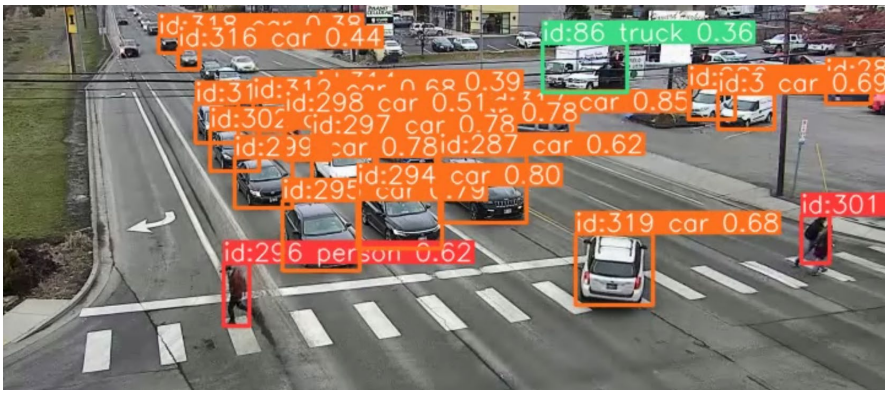


Figure 3.12 Conflicted zone during turning movement

Figure 3.13 Conflicted zone due to stop bar encroachment.

Once again, a stop bar encroachment does not necessarily indicate an intersection to be dangerous. However, frequent occurrences may reveal a pattern of inattentive driving or disregard for pedestrian crossings. Furthermore, on a day with poor weather or in a car with poor brakes, perhaps this behavior translates to a pedestrian being hit in the crosswalk. However, this metric is imperfect, as the insights largely depend on drawing zones that can provide helpful information about the interactions between vulnerable users and cars. For example, Figure 3.14 shows a situation that the research team would consider to be relatively benign. However, the script would flag it as a conflicted zone incident if the entire crosswalk were defined as a zone.

Figure 3.14 Example of benign conflict.

In Figure 3.14 there are two full lanes of separation between the SUV and the pedestrian, and the SUV is well within the law to turn right on red after arriving at a complete stop. Situations such as these highlight the nuances in traffic conflicts and the difficulties in successfully separating dangerous events from safe ones. The hope is that given a long enough duration of tracked users, these nuances might be less important, and the intersection conflicted zone exposure metric might be a useful tool in discerning which intersections have a higher incidence of interactions between pedestrians/bicyclists and cars within crosswalks. Table 3.4 is the conflicted zone exposure results for an hour at Camera A.

Table 3.4 Pedestrian conflicted zone exposure

Pedestrian ID	Number of Frames in Zone	Number of Frames in Conflict	Conflicted Zone Exposure
212	240	0	0.00
213	3	0	0.00
214	399	0	0.00
610	239	58	24.27
735	282	12	4.26
782	124	0	0.00
841	154	154	100.00
952	238	0	0.00
957	22	0	0.00
958	159	0	0.00
1235	6	0	0.00
1238	21	0	0.00

Pedestrian ID	Number of Frames in Zone	Number of Frames in Conflict	Conflicted Zone Exposure
1245	208	0	0.00
1406	260	25	9.62
1459	273	0	0.00

Table 3.4 shows that vulnerable roadway users were detected within the zone for between less than a second and 14 seconds. Most could cross the street without conflicted zone exposure. However, a few were in conflict with a vehicle, and one user was in conflict for their 5-second crossing. This was the result of a severe stop-bar encroachment, as seen in Figure 3.15. The intersection conflicted zone exposure score was 9.47, i.e., 249 conflicted frames/2628 frames * 100. It is unclear at this point what values of conflicted zone exposure are dangerous and which ones are not worrisome; however, given enough video at enough locations, a relationship could potentially be developed between this value and observed crashes, or compared across intersections to determine relative danger.

Figure 3.15 Example of conflicted zone exposure.

3.6. Summary

This chapter describes the framework used to perform open-source computer vision analysis and demonstrated potential use cases available to engineers and researchers. The first case study demonstrated the ability to monitor traffic with open-source computer vision technology. The case study also acknowledged some potential insights that could be gleaned by spatially or temporally comparing user counts. The second case study demonstrated the system used to produce desire line heat maps, which help visualize the usage of facilities and detect potentially dangerous trends. The third case study introduced and described the potential

usefulness of a new surrogate safety measure for pedestrians and bicyclists at intersections. The case study looked at the situations that constitute a safety concern and those that do not, and it explained the goal of distinguishing between them.

Some drawbacks associated with open-source computer vision deployments have already been addressed, but they are summarized them here. Open-source detection methods haven't achieved acceptable levels of accuracy, as they still produce a significant number of misidentifications and missed identifications, making it difficult to trust the results obtained. Next, the systems are highly resource intensive in regard to storage space and computational power, so it is simply infeasible to conduct large-scale analysis such as the 24/7 analysis that can be done with proprietary systems. Another concern is that significant troubleshooting must be done to write the code for an analysis system, and because of the rapidly changing landscape of this field, things can quickly become out of date or require modification. Lastly, the video sources that are being used were not set up with computer vision analyses in mind. This fact has meant that the cameras were not necessarily located where we would have chosen for our analysis but were instead in locations that are helpful to ITD in observing traffic situations. Additionally, the camera's view fields are occasionally moved by ITD to accommodate its monitoring needs, which means that the previously defined zone locations are no longer relevant to the new configuration.

With that being said, open-source computer vision has significant advantages. First, the technology is free, as Python is an open-source coding language, and the YOLO package is published online. Second, the possible analysis locations are limited only by the camera feeds that one can access. This means that a diverse set of intersections can be observed, and we were fortunate to partner with ITD to receive additional video sources from its existing camera network. Third, the technology is highly customizable, and the results are less aggregated than those of proprietary sensors, which presents more opportunities for analyses, including user-level surrogate safety analysis.

CHAPTER 4. CONCLUSIONS AND FUTURE RESEARCH

4.1. Conclusions

This research revealed the value of computer vision for traffic monitoring and safety analysis. We demonstrated how our computer vision code can accomplish essential traffic monitoring tasks such as volume counts. Moreover, unlike traditional methods, computer vision can create desire-line heat maps. Finally, we used our computer vision code to develop a new safety measure called Conflicted Zone Exposure that can be used to identify dangerous roadways. We also identified some of the benefits and drawbacks of using open-source computer vision technology.

4.2. Future Research Opportunities

This project demonstrated the exciting opportunities for computer vision. However, significant advancements are still yet to be made in this field, which should be researched further. The first opportunity is the ability to develop spatial relationships between objects by using a perspective transformation matrix. More information on this technique can be found in Appendix A. Constant advancements are happening in computer vision, so it is difficult to foresee what advancements will happen in the future to overcome computing speed and detection accuracy issues. There is certainly potential for research that further investigates the relationship between model size and accuracy to validate counts by using different model sizes and to quantify corresponding detection time increases.

Lastly, the conflicted zone exposure metric bears further exploration to determine whether there is a correlation with areas that have high crash rates involving pedestrians and bicyclists. This study would require computing resources to analyze significantly more footage, but it could be completed using the existing tools that were developed in this research.

REFERENCES

Allen, B. L., Shin, B. T., and Cooper, P. J. (1978). Analysis of traffic conflicts and collisions (No. HS-025 846).

Bottero, M., Dalla Chiara, B., and Deflorio, F. P. (2013). Wireless sensor networks for traffic monitoring in a logistic centre. Transportation Research Part C: Emerging Technologies, 26, 99-124.

Bullock, D., Grenard, J., and Tarko, A. (2002). Evaluation of selected video detection systems at signalized intersections. *Joint Transportation Research Program, 91*.

Cynecki, M. J. (1980). Development of a conflicts analysis technique for pedestrian crossings. Transportation Research Record, 743, 12-20.

Dilek, E., and Dener, M. (2023). Computer vision applications in intelligent transportation systems: a survey. *Sensors*, *23*(6), 2938.

Espinoza, F. T., Gabriel, B. G., and Barros, M. J. (2017, October). Computer vision classifier and platform for automatic counting: more than cars. In *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)* (pp. 1-6). IEEE.

FHWA. (2022, December). 2022 Traffic Monitoring Guide - Federal Highway Administration.

Fontaine, M., Olariu, S., and Weigle, M. C. (2009). Traffic monitoring. Vehicular Networks from Theory to Practice.

Gettman, D., and Head, L. (2003). Surrogate safety measures from traffic simulation models. *Transportation Research Record*, *1840*(1), 104-115.

Hydén, C. (1987). The development of a method for traffic safety evaluation: The Swedish Traffic Conflicts Technique. *Bulletin Lund Institute of Technology, Department*, (70).

Jones, S. (2023, September 6). *The Problems With Police Crash Reports*. Strong Towns.

Karagulian, F., Liberto, C., Corazza, M., Valenti, G., Dumitru, A., and Nigro, M. (2023). Pedestrian Flows Characterization and Estimation with Computer Vision Techniques. Urban Science, 7(2), 65.

Khattak, K. S., Minallah, N., Khan, Z. H., Khan, M. A., Khan, A. N., and Ayaz, S. (2022). Sensing technologies for traffic flow characterization: From heterogeneous traffic perspective. Journal of Applied Engineering Science, 20(1), 29-40.

Klebelsberg, D. (1964). Derzeitiger sand der verhaltensanalyse des kraftfahrens. Zrbeit und leitsung. *Ablt. Arbeitswissenscaft soziale betriebspraxis*, *18*, 33-37.

Lee, K., and Sener, I. N. (2020). Emerging data for pedestrian and bicycle monitoring: Sources and applications. Transportation research interdisciplinary perspectives, 4, 100095.

Li, J., Hajimirsadeghi, H., Zaki, M. H., Mori, G., & Sayed, T. (2014). Computer vision techniques to collect helmet-wearing data on cyclists. *Transportation Research Record*, *2468*(1), 1-10.

Molina, J. M. G., Sotis, A. S., Pascual, R. C., Cubillas, J. E. D., and Matias, J. B. (2022, December). Development of Real-Time Traffic Monitoring and Visualization System Using Stationary Roadside Sensor. In Proceedings of the 2022 11th International Conference on Networks, Communication and Computing (pp. 154-160).

Moro, A., Mumolo, E., Nolich, M., and Umeda, K. (2011, October). Real-time GPU implementation of an improved cars, pedestrians and bicycles detection and classification system. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (pp. 1343-1348). IEEE.

National Center for Statistics and Analysis. (2023a, June). Bicyclists and other bicyclists: 2021 data (Traffic Safety Facts. Report No. DOT HS 813 484). National Highway Traffic Safety Administration.

National Center for Statistics and Analysis. (2023b, June). Pedestrians: 2021 data (Traffic Safety Facts. Report No. DOT HS 813 458). National Highway Traffic Safety Administration.

Pourhomayoun, M. "Automatic Traffic Monitoring and Management for Pedestrian and Cyclist Safety Using Deep Learning and Artificial Intelligence" Mineta Transportation Institute Publications (2020).

Rogers, S., and Papanikolopoulos, N. P. (2000, October). Counting bicycles using computer vision. In *ITSC2000. 2000 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 00TH8493)* (pp. 33-38). IEEE.

Sayed, T., Zaki, M. H., and Autey, J. (2013). Automated safety diagnosis of vehicle–bicycle interactions using computer vision analysis. *Safety science*, *59*, 163-172.

Smaldone, S., Tonde, C., Ananthanarayanan, V., Elgammal, A., and Iftode, L. (2010). *Improving bicycle safety through automated real-time vehicle detection*. Rutgers University.

Takahashi, K., Kuriya, Y., and Morie, T. (2010, November). Bicycle detection using pedaling movement by spatiotemporal gabor filtering. In *TENCON 2010-2010 IEEE Region 10 Conference* (pp. 918-922). IEEE.

Tarko, A. P. (2012). Use of crash surrogates and exceedance statistics to estimate road safety. *Accident Analysis & Prevention*, *45*, 230-240.

Ultralytics (2023) https://www.ultralytics.com/blog/exploring-yolo-vision-2023-a-panel-talk-overview

Wong, P. K. Y., Luo, H., Wang, M., Leung, P. H., and Cheng, J. C. (2021). Recognition of pedestrian trajectories and attributes with computer vision and deep learning techniques. Advanced Engineering Informatics, 49, 101356.

Xu, W., Ruiz-Juri, N., Huang, R., Duthie, J., and Clary, J. (2018, June). Automated pedestrian safety analysis using data from traffic monitoring cameras. In *Proceedings of the 1st ACM/EIGSCC Symposium on Smart Cities and Communities* (pp. 1-8).

Zaki, M. H., Sayed, T., Tageldin, A., and Hussein, M. (2013). Application of computer vision to diagnosis of pedestrian safety issues. Transportation research record, 2393(1), 75-84.

APPENDIX A: TOOLS FOR ACCESSING DIFFERENT TYPES OF VIDEO FEEDS AND FURTHER ANALYSIS

A.1. YouTube Live Streams

We accessed YouTube Live Streams via a Python package called pafy. We created a function to create the data stream for recording shown in Figure A.1.

```
def create_yt_folder_and_stream(example_data_folder, camera_name, youtube_url):
    """Creates folder and stream for youtube urls."""

    # Create camera folders
    camera_folder = os.path.join(example_data_folder, camera_name)
    if not os.path.exists(camera_folder):
        os.makedirs(camera_folder)
        # Creates a text file with information about the camera
        info_txt_file = camera_folder + r'\info.txt'
        with open(info_txt_file, 'w') as f:
            f.write(str("Camera name:")  + '\n')
            f.write(str(camera_name) + "\n") # camera name
            f.write(str("The url address: ") + '\n' )
            f.write(str(youtube_url) + '\n') # camera district and id number
            f.write(str(time.time()))

    resource_images = os.path.join(camera_folder, "resource_images")
    if not os.path.exists(resource_images):
        os.makedirs(resource_images) # Creates the new image folder

    # pafy version
    play = pafy.new(youtube_url).getbest()
    assert play is not None # we want to make sure their is an input to read.
    stream = play.url


    return camera_folder, stream
```

Figure A.1 YouTube stream function.

The inputs for this function are the path of the folder to which you wish to write your data, a camera name that does not have to be anything specific but is simply how you would like to identify the camera (the shortest name that allows you to recognize the camera is generally best, and as a standard practice, we usually chose the name of the city in which the camera was located), and finally the URL for the camera's feed on YouTube. The function takes these inputs and creates a directory to save data to within your designated folder. Next it documents the name of the camera, its URL, and the start time of the stream within a txt file. Then it creates the stream object, which is necessary for performing analysis with YOLOv8. From here, the information can be passed to our standard analysis pipeline, which is used for all video sources. This pipeline is displayed in the function in Figure A.2, named main.

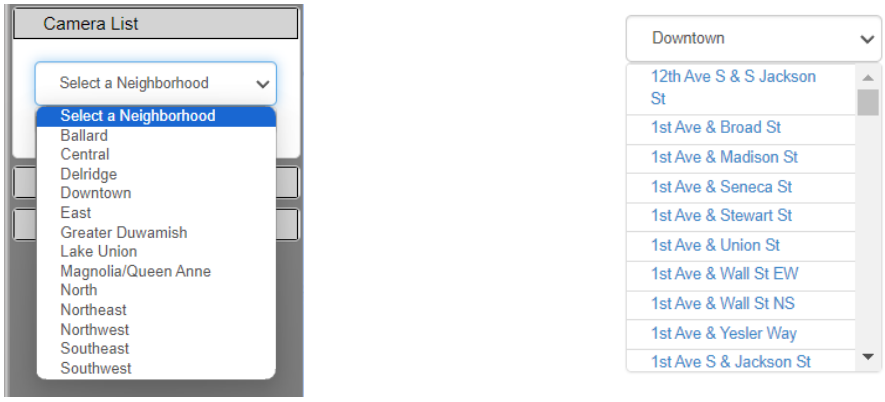
```python
def main(camera_folder, stream, recording_duration, display_while_recording, video_clip_users, tracker_stride_rate, desire_line_users):
    """Combines everything."""

    recording_folder = record_stream(camera_folder, stream, recording_duration, display_while_recording)
    df_tracker = create_tracker(recording_folder, video_clip_users, tracker_stride_rate)

    df_zone_occupancy = get_zone_occupancy(recording_folder, df=df_tracker)
    df_CZE = get_zone_conflict(recording_folder, df=df_zone_occupancy)
    create_heatmap(recording_folder, df=df_tracker, desire_line_types=desire_line_users)


    return df_CZE
```

Figure A.2 Main computer vision function.

This function takes the stream object created in the previous function and performs the following tasks: First a video is recorded for the specified duration next. Next, objects are tracked to produce a tracker CSV file that contains object classes and locations. Next, the tracker filter is referenced against specified zones to create a zone occupancy CSV. Then the conflicted zone exposure is calculated and saved to a CSV. Finally, a heat map showing the paths that the object took is plotted and saved to the folder directory.

A.2. Video Scraping

Obtaining footage from cameras in the City of Seattle required a different process because while the cameras are publicly available, there is no structure in plain for accessing the feed through Python. This required implementing video scraping, which relies on a Python package called Selenium, which serves as an intermediary between Python and a web browser. When a web browser plays videos, the videos are briefly saved in a media log in the form of ts files, a video transport stream (ts) file. These ts files are approximately 10-second clips that can be downloaded during a limited time window. The scraping technique implemented in our Python code allows us to open a browser, play the video for 30 seconds, look for ts files to download, close the browser, and repeat the process to access the next set of ts files. The inputs for this process are the duration of the video to be collected, as well as a neighborhood and a camera number. Figures A.3 and A.4 show the neighborhood and camera selection boxes from the Seattle Department of Transportation website.



(a) (b)

Figure A.3 Seattle camera selection menus.

The scraping process works functionally as if a person were manually navigating the website and clicking on the boxes, but automating the process allows it to be sped up and made viable for collecting longer video durations. The neighborhood input tells the Selenium package which neighborhoods to click on, and then the camera number tells it which item in the list of cameras for the given neighborhood should be clicked on. The result is a series of short clips that can be merged into a single .mp4 file fed to the computer vision analysis functions described above.

A.3. RTSP Address

The final method for accessing video was through Real Time Streaming Protocol (RTSP) streams. These are unique addresses for each of ITD's cameras, which can be accessed through a Python function while logged onto ITD's server via VPN. RTSP was first developed in 1996-1997 by a team comprising of RealNetworks, Netscape, and Columbia University, and it remains the method of choice for distributing most closed-circuit television cameras today (Henken, 2024). We developed a function specifically for accessing ITD RTSP streams through Python for analysis with YOLOv8. This function is displayed in Figure A.4.

```python
def create_itd_folder_and_stream(example_data_folder, itd_district, camera_number):
    """Creates folder and stream for ITD cameras."""

    # Obtaining Camera Information.
    camera_workbook = example_data_folder + r'\ITD_Cameras.xlsx' # This is the key that shows details for all cameras.
    df_cameras = pd.read_excel(camera_workbook)
    camera_specs = df_cameras.loc[(df_cameras['ID'] == camera_number) & (df_cameras['District'] == itd_district)] # This filters down to
    lat = camera_specs.iloc[0]['Latitude'] # Camera latitude
    lon = camera_specs.iloc[0]['Longitude'] # Camera longitude
    IP = camera_specs.iloc[0]['IP'] # Unique camera ip address
    location = camera_specs.iloc[0]['Location'] # Camera locations (i.e. crossroads)

    # Create camera folders
    camera_name = "ITD_" + str(itd_district) + "-" + str(camera_number)
    camera_folder = os.path.join(example_data_folder, camera_name)
    if not os.path.exists(camera_folder):
        os.makedirs(camera_folder)

        info_txt_file = camera_folder + r'\info.txt' # Creates a text file with information about the camera
        with open(info_txt_file, 'w') as f:
            f.write(str("Camera name:")  + '\n')
            f.write(str(camera_name) + "\n") # came name
            f.write(str("Camera location:")  + '\n')
            f.write(str(location) + "\n") # camera location
            f.write(str("The camera district and id number: ") + '\n' )
            f.write(str(itd_district)+ r'-' + str(camera_number)+ '\n') # camera district and id number
            f.write(str("Latidude and Longitude: ") + '\n' )
            f.write(str(lat)+ "," + str(lon) + '\n')

    resource_images = os.path.join(camera_folder, "resource_images")
    if not os.path.exists(resource_images):
        os.makedirs(resource_images) # Creates the new image folder

    stream = r'rtsp://' + str(IP) + r'/axis-media/media.amp' # This is default resolution.

    os.environ['OPENCV_FFMPEG_CAPTURE_OPTIONS'] = 'rtsp_transport;udp' # Use tcp instead of udp if stream is unstable

    return camera_folder, stream
```

Figure A.4 ITD camera access function.

Each camera's RTSP address and location are stored in an Excel file. The function performs three tasks: First, it creates the file location where this camera's data will be stored. Next, it writes information about that camera to a txt file for later reference. Lastly, it formats the RTSP address to create the stream object, which is then passed to the main CV analysis function described above.

A.4. Perspective Transformation

Perspective transformation is an important advancement in this field which was investigated but not fully accomplished. The general idea is that a matrix develops a relationship between the image coordinates and real-world planar coordinates. This relationship is shown in Figure A.5.

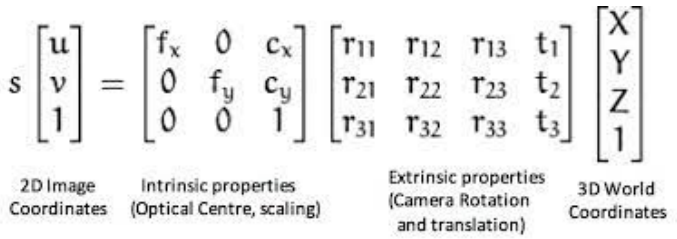


Figure A.5 Perspective transformation formulation. *Source*: Intel, 2019

The system can be simplified by approximating the roadway surface as a plane with a constant z coordinate of zero. A transformation matrix (the product of the intrinsic and extrinsic matrix) can be solved by six corresponding pairs of image coordinates and real-world coordinates. Tests of this method proved successful at a small scale. However, they could not be scaled up to an intersection for this study, given the image resolution and necessary precision required for obtaining measurements that provide the relationship between these points. The development of a system that could reliably transform the coordinates of detected objects into planar coordinates would be beneficial for the purposes of safety analysis, as a conflict between a pedestrian or bicyclist and a car could be determined on the basis of a measure of distance rather than spatial proximity being dictated by a point in polygon test. This would enable a better measure of near-miss events and the development of a relationship between crashes and near-miss events. Another advantage would be the ability to perform speed estimation given known distances and time stamps, which could inform researchers about how a roadway is being used and general human behaviors such as an updated estimation of pedestrian walking speed.

A.5 References

Henken, P. (2024, February 29). *RTSP – all you need to know about real-time streaming protocol* . Kaltura Blog.

Intel. (2019, December 31). *Camera calibration and 3D reconstruction*. Camera Calibration and 3D Reconstruction - OpenCV 2.4.13.7 documentation.