12-11-2023

Perception Intelligence Integrated Vehicle-to-Vehicle Optical Camera Communication.

Khadija Ashraf
Georgia State University

Perception Intelligence Integrated Vehicle-to-Vehicle Optical Camera Communication

by

Khadija Ashraf

Under the Direction of Ashwin Ashok, Ph.D.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2023

ABSTRACT

Ubiquitous usage of cameras and LEDs in modern road and aerial vehicles open up endless opportunities for novel applications in intelligent machine navigation, communication, and networking. To this end, in this thesis work, we hypothesize the benefit of dual-mode usage of vehicular built-in cameras through novel machine perception capabilities combined with optical camera communication (OCC). Current key conception of understanding a line-of-sight (LOS) scenery is from the aspect of object, event, and road situation detection. However, the idea of blending the non-line-of-sight (NLOS) information with the LOS information to achieve a see-through vision virtually is new. This improves the assistive driving performance by enabling a machine to see beyond occlusion. Another aspect of OCC in the vehicular setup is to understand the nature of mobility and its impact on the optical communication channel quality. The research questions gathered from both the car-car mobility modelling, and evaluating a working setup of OCC communication channel can also be inherited to aerial vehicular situations like drone-drone OCC. The aim of this thesis is to answer the research questions along these new application domains, particularly, (i) how to enable a virtual see-through perception in the car assisting system that alerts the human driver about the visible and invisible critical driving events to help drive more safely, (ii) how transmitter-receiver cars behaves while in the mobility and the overall channel performance of OCC in motion modality, (iii) how to help rescue lost Unmanned Aerial Vehicles (UAVs) through coordinated localization with fusion of OCC and WiFi, (iv) how to model and simulate an in-field drone swarm operation experience to design and validate UAV coordinated localization for group of positioning distressed drones. In this regard, in this thesis, we present the end-to-end system design, proposed novel algorithms to solve the challenges in applying such a system, and evaluation results through experimentation and/or simulation.

INDEX WORDS: NLOS Perception, See-through-a-vehicle, LED-Camera Communication, Vehicular Motion Modelling, Drone-Drone localization, Drone aided localization, Drone positioning modelling and simulation, Vehicular VLC.

Perception Intelligence Integrated Vehicle-to-Vehicle Optical Camera Communication

by

Khadija Ashraf

Committee Chair: Ashwin Ashok

Committee: Xiaojun Cao

Rajshekhar Sunderraman

Shubham Jain

Electronic Version Approved:

Office of Graduate Services

College of Arts and Sciences

Georgia State University

December 2023

DEDICATION

I dedicate my dissertation to my husband Mahbub, my kids Umar, and Khalid. I'm so grateful to almighty for having them in my life. Their love, patience, and trust for me is the fuel of my life.

ACKNOWLEDGMENTS

All praise to Almighty Allah (swt) the Most Gracious and the Most Merciful, for His blessings given to me during my study and in completion of this dissertation.

I would like to sincerely thank my advisor Dr. Ashwin Ashok for constantly advising me on my research with interesting ideas, challenges, and guidance. He has been a kind and compassionate mentor who is always there to help. I'm thankful to my co-workers in the MORSE Studio Lab. My special thanks to Dr. Rajshekhar Sunderraman, Dr. Xiaojun Cao, and Dr. Shubham Jain for their valuable feedback in my research.

This research has been supported by the National Science Foundation (NSF) under Grant Nos. CNS-1901133 and CNS-2000475, and through the GSU Provost Doctoral Dissertation Award (July 2021-Apr 2022).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Multiple lenses in a stereo camera allows the camera to simulate human binocular vision and therefore gives it the ability to perceive depth along with the horizontal and vertical dimensions. Moreover, in the Optical Camera Communication (OCC) using demodulation techniques a camera image sensor is capable of receiving optical signal from optical transmitting sources (i.e., Light Emitting Diode). The dual usage of stereo camera simultaneously as a depth scene viewer and as a data receiver creates remarkable opportunities for research in the vehicular optical stereo camera communication field. Many challenging areas like vehicle localization, road environment perception, driving safety-critical event recommendation, etc., are being enriched by the community researcher leading to partial/full automated driving.

For a vehicle understanding the surrounding and communicating the perceived information to neighboring vehicles in the native V2V network is one of the key components in partially/fully automated navigation system in the current time. However, a successful implementation of scene perception, and information transmission is challenged by factors like, smart machine vision, mobility, high data-rate, spectrum scarcity, transmission collision, timely-responses (time critical), and so on. We implement an intelligent scene perception component, and low data-rate communication using LEDs as transmitter and the duality of stereo camera as optical signal receiver and scene perceiver. We also model the motion and the movement behaviour of transmitting vehicle relative to the receiving vehicle in car-car

OCC setup. We further argue that, aerial vehicular (drones) area has many overlapped struggles with the road vehicles. The on-board cameras and light sources already available in drones can be used to find solutions to critical problems like short-range, and long-range lost drone rescue mission by leveraging OCC.

The NLOS perception enables a machine to see beyond occlusion. In our research we present the design, implementation, and evaluation of non-line-of-sight (NLOS) perception to achieve a virtual see-through functionality for road vehicles. In this system, a safety event, such as pedestrian crossing or traffic light status or vehicle merge, that are occluded to a driver due to another vehicle on the driving lane, are perceived and communicated by the occluding vehicle's LED transmitters. In our prototype implementation each vehicle is equipped with a camera, placed on the dashboard, that perceives the scene in a drivers view. This scene is analyzed and mapped into specific warning codes that are specific to a safety event, and communicated as short packets using visible light communication. We extend our base system implementation of NLOS further, to design and validate a multiple access setup with single receiver vehicle and multiple transmitter vehicles. We position the design and use-cases of a cognitive information processing model pipeline for driving assistance that generates spontaneous safety alerts by understanding and prioritizing surrounding LOS and NLOS information from multiple transmitting vehicles. We designed a gist recognition algorithm comprising of prediction-attention concepts. We narrate our NLOS perception story in the chapter 2. As our research focuses the OCC and its application in the vehicles, we put an effort to characterize the motion modality of road vehicles in OCC. The amount of motion

of vehicles in real world vehicular OCC use–case scenarios has not been explored prior. In this work we present a mobility characterization study through extensive experiments in real world driving scenarios. We characterize motion using a constantly illuminated transmitter on a lead vehicle and a multi–camera setup on a following vehicle. The observations from our experiments reveal key insights on the degree of relative motion of a vehicle along its spatial axis and different vehicular motion behaviors. We take a further step ahead and build a ground truth enabled depth estimating stereo camera. Our receiver camera's hardware setup is integrated with the RaspberryPi module and controlled remotely through secure shell (SSH) and to operate (capture image and video) synchronously. A single channel 25 meters range radius, and 16000 samples per second enabled LIDAR is used for ground truth distance measurements. The details of our motion characterization work in imprinted in the chapter 3. Next, we invest our acquired knowledge of scene perception, intelligent driving recommendation generation, and motion characterization from the road-vehicle OCC to the aerial vehicles.

We propose an alternative solution to GPS redundancy for multi drone systems. We propose a design where drones with no GPS or non-functional GPS units are able to receive help to localize themselves coordinated by a helper drone with a functional GPS/location sensing unit. The key concept of our methods is a *helper drone* (HD) tracks nearby GPS *distressed drones* (DD), and then estimates the relative position of the distress drone – by estimating the distance and bearing angle. In our preliminary baseline, we implemented and prototyped an ACK based P2P WiFi communication to conduct inter-drone data transfer between two

drones. Next, in our P2PDroneLoc work, we evaluate two fundamental approaches for range estimation for peer-to-peer relative positioning between drones: (a) camera and computer vision projection theory, and (b) WiFi Fine Time Measurements (FTM). Next, through design and implementation of the drone swarm localization model we complete our drone localization endeavor. Moreover, we design model simulations using parameter sweeping with control parameters like average localization error, swarm of DDs. We present an in-depth validation of our previously innovated (i) camera localization, (ii) WiFi localization methods. Through simulated experiment we manifested that our drone swarm localization simulation results are in harmony with the in-field camera and WiFi experiment results. Our Model allows to curate a real-life coordinated drone swarm localization environment setup, provides flexibility to design several simulations with parameter sweeping for significant control parameters like any real-world drone swarm setup would require and leverage. We confirm through our result evaluation that our innovative drone coordinated camera and WiFi localization protocols are not only universally usable, also establish a pioneering research opportunity in the drone localization, to be discussed in chapter 4.

Thesis Statement: Stereo camera perception can be used to establish non-line-of-sight perception and bandwidth-efficient data transmission in road and aerial vehicular communications through vehicular localization and motion modeling, intelligent scene understanding, and relaying the inferred scene information to the neighboring vehicles in the network.

1.1 Part A: Car-Car Stereo Camera Communication

1.1.1 Part A1: Non-Line-Of-Sight (NLOS) Perception

The ability to perceive safety-critical events by virtually seeing through vehicles and other obstructions on the road can be a very useful driving assistance feature for vehicles. We first hypothesise that such a feature can be achieved by vehicles driving ahead proactively communicate information about the visual scenery they perceive using dashboard cameras. Considering brake-light light emitting diode (LED) to camera communication as the enabler, we present the design, implementation, and evaluation of a physical prototype of non-line-of-sight (NLOS) perception to achieve a virtual see-through functionality for road vehicles in the single access setting titled See-through a Vehicle (2).

In this See-through a Vehicle system, a safety event, such as pedestrian crossing or traffic light status or vehicle merge, that are occluded to a driver due to another vehicle on the driving lane, are perceived and communicated by the occluding vehicle. Each vehicle is equipped with a camera, placed on the dashboard, that perceives the scene in a drivers view. This scene is analyzed and mapped into specific warning codes that are specific to a safety event, and communicated as short packets using visible light communication. The camera in the following vehicle captures this information and generates a recommendation for safety action to the driver by comparing the warning from the packet and from its own scene perception. Through experimental evaluations of a proof-of-concept implementation, we show that our system is able to achieve up to 90% accuracy in identifying nine occluded safety events, which correspond to traffic light statuses (red, green, yellow), other vehicles'

lane change behaviors (merge/leave lane left/right), and pedestrian detection. We also made our dataset (3) publicly available with an intention of serving the community.

We further extend the virtual see through functionally implementation for multiple access setting with single receiver and multiple transmitters. In this paper, we position an information processing pipeline for predicting the non-line-of-sight safety event. In particular, we present the algorithmic design of the cognitive information processing system that will continuously warn the host driver about the line-of-sight and non-line-sight road situations. This system hosts a (i) prediction-attention module that inherits the concept of gist recognition by linking position and interactions between individual road objects from a sequence of locally perceived camera scene data, and a (ii) decision-making module that makes an informed decision to choose the ultimate safety event by prioritising among the chosen LOS event and received non-line-of-sight (NLOS) events from neighbouring vehicles. We position the use-case of our proposed information processing pipeline through a case-study analysis for three real-world driving scenarios.

1.1.2 Part A2: Car motion analysis

The mobility of the vehicles presents a fundamental impediment for high throughput and link sustenance in vehicular Optical Camera Communication(OCC). While prior work has explored vehicular OCC system design, yet, there is no clear understanding on the amount of motion of vehicles in real world vehicular OCC use–case scenarios. To address this knowledge gap, we present a mobility characterization study (1) through extensive experiments in real world driving scenarios. We characterize motion using a constantly illuminated transmitter

on a lead vehicle and multi–camera setup on a following vehicle. The observations from our experiments reveal key insights on the degree of (relative) motion of a vehicle in each of its 3D axis and the different vehicular motion behaviors. The motion characterization from this work lays a stepping stone to addressing mobility in vehicular VLC through sophisticated tracking and link sustenance protocols.

We further enrich our motion characterization (1) prototype in the next project where we built a receiver stereo camera setup with two RaspberryPis and the corresponding analysis of the depth (distance between vehicles) related information. This new stereo camera receiver setup features two RaspberryPi cameras. The cameras, integrated with the RaspberryPi module, were controlled remotely through secure shell (SSH) and to operate (capture image and video) synchronously. A SLAMTEC Rplidar A3 is embedded in the custom built camera for ground truth distance measurements, which is a single channel, 25 meters range radius, 16000 samples per second enabled LIDAR. Considering cameras as our measuring unit at the receiver, we denote the horizontal (X dimension: δ_u) and vertical (Y dimension: δ_v) motion parameters in pixel units (1). Additionally, the motion along the Z dimension, or depth, is measured as the spatial separation of the transmitter and receiver cars at a given time snapshot, which is equivalent to the estimating the distance between the two vehicles at each time instance through stereo depth estimation. Overall, we collected measurements worth of about 12000 image frames and over 10000 sensor data samples. The motion model are derived from this measurement dataset by using tools from computer vision, probability and statistical error analysis.

1.2 Part B: Drone Coordinated Localization

In our above mentioned works in Part A 1.1 we have already implemented single and multiple access vehicular optical stereo camera communication channel, intelligent packet communication mechanism for NLOS perception to help communicate vehicles with each other. While setting up the OCC channel we addressed fundamental challenges affixed with such a line-of-sight wireless communication modality when mobility comes into the scene. Now that, we have the understanding of motion modality and communication intricacies we apply it in the aerial communication, particularly in UAV communication. In this work we design, implement, and evaluate a drone assisted coordinated localization system to help navigate remotely operating Global Positioning System (GPS) denied drones. Without any requirements of a base station or a central entity, in this system, a GPS enabled *helper drone* offers localization help to low functionality *distress drones*. We divide our entire project blueprint into two phases, (i) Peer-to-Peer Localization for GPS-Denied Drones (ii) Modelling and Simulation of Drone Swarm Localization.

1.2.1 Part B1: Peer-to-Peer Localization using camera and WiFi FTM

In the localization phase, we propose a solution to achieve coordinated localization between two unmanned aerial vehicles (UAVs) using radio and camera based optical wireless communication. We propose to achieve the localization between the UAVs to address the problem of UAV GPS failure or its unavailability. Our proposed approach allows one UAV with a functional global positioning system (GPS) unit to coordinate the localization of another

UAV with a compromised or missing GPS system. Our solution for localization uses a sensor fusion and coordinated wireless communication approach. In this work, helper drone when discovers a distressed drone flying in proximity, estimates the distress drone's absolute GPS location information respective to its own location. This is performed by computing the relative position of the distress drone with the helper drone. The estimated location is then communicated to the distress drone. In this paper, we evaluate two fundamental approaches for range estimation for peer-to-peer relative positioning between drones: (a) camera and computer vision projection theory, and (b) WiFi Fine Time Measurements (FTM). We envision that a helper drone equips a camera and a WiFi FTM access point (AP), while a distress drone equips only a WiFi FTM AP. We evaluate our proposed peer-to-peer localization via accuracy across three estimated measures: (i) range or distance between the drones, (ii) GPS bearing (angle), and (iii) GPS location (coordinates). Based on our analysis of our dataset consisting of outdoor static and flying drones setup, our methods result in a median localization accuracy within 1-4m.

1.2.2 Part B2: Modelling and Simulation of Drone Swarm Localization

We also model and simulate drone swarm localization with several distressed drones to evaluate our camera and WiFi FTM localization methods. The drone swarm modelling allows us to conserve the inherent nature of random motion and interactions of the real-world in-field experimentation while measuring the average localization error, and average execution time in the presence of an increased number of distressed drones. We conducted up to 160 simulations for 8 different parameter combinations. The simulation results attest

our in-field experimental localization error ranges, and give us a holistic idea of the end-end operation-conduct timing for different number of drone participants.

CHAPTER 2

NON-LINE OF SIGHT PERCEPTION: ACHIEVING VIRTUAL SEE-THROUGH ABILITY IN VEHICLES USING INTELLIGENT PERCEPTION AND OPTICAL CAMERA COMMUNICATION IN BOTH SINGLE AND MULTIPLE ACCESS

It is well documented that distracted driving (4) and reckless driving (5) are the major causes of road accidents and fatalities, and such situations are largely attributed to the specific activities of a driver (e.g. looking at a phone or over-speeding). In addition to the typical causes of accidents due to driver distraction, the possibility of accidents due to obstructions on the driving path or roadway cannot be ignored. In fact, according to a national motor vehicle crash causation survey (6) conducted by the US National Highway Traffic Safety Administration (NHTSA), obstruction to the driver accounts for about 36% of the vehicle crashes – among 5471 crashes recorded over 3 years from 2005-2008. One of the key reasons for these obstructions are the other vehicles in vicinity. Obstructions and occlusions due to other vehicles on the road result in the lack of information about any activity on the driving path beyond the field of view (FOV) of the driver. Thus, when the driver is confronted with an emergency scenario, due to the lack of information about what is to come ahead and the extremely short time to react, the situation leads to an accident.

To address the safety-critical scenarios created due to roadway obstructions, in this work, we position the idea of extending the driver's perception of the scene beyond line-of-sight (LOS), or non-line-of-sight (NLOS) perception. The key notion is to provide a virtual see-through ability to perceive the scenarios beyond LOS within the driver's FOV. In this way, the driver is always informed of what event/scenario may be coming ahead, and provide

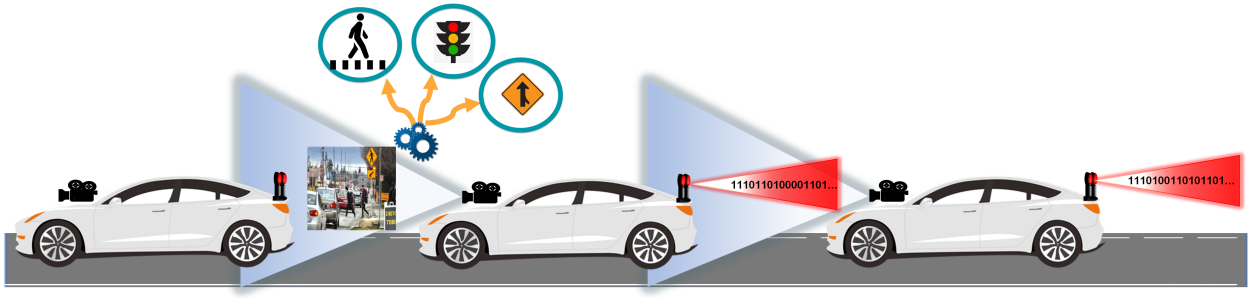


Figure 2.1: A conceptual diagram of the proposed non-line-of-sight (NLOS) perception system on vehicles using computer vision perception and LED-Camera communication.

sufficient time to plan a course of action to confront those events or scenarios.

2.1 See-through a Vehicle: Augmenting Road Safety Information

To realize NLOS perception in practice, in this paper, we propose a novel technique that leverages the advancements in state-of-the-art deep neural networks (DNN), computer vision and visible light camera communication. The proposed technique involves communicating the information about the occluded scene to the driver's vehicle by the occluding vehicle itself. In this regard, we design, implement and evaluate a vehicular NLOS perception system that uses off-the-shelf cameras to perceive the occluded scene and communicate that information to the driver's vehicle using camera (optical wireless) communication. The key idea is to generate a highly compressed and abstracted information from (camera) scene analysis, and communicate that information from one vehicle to another using a line-of-sight link. By intelligently combining or fusing this information with the vehicle's own camera scene analysis, each vehicle will be able to intelligently infer and generate appropriate driving/safety recommendations to the driver/vehicle.

As illustrated in Fig. 2.1, the proposed approach realizes NLOS perception using scene

perception from a stereo-camera fit onto the front of every vehicle and doubling the brake-light light emitting diodes (LED) of the vehicle as a visible light communication (VLC) transmitter. The scene analysis generated from the stereo-camera perception is transmitted by the brake-light light emitting diodes (LED) by modulating the information bits on to its optical intensity (IM or intensity modulation). The camera on the (receiver) vehicle driving behind captures the optically modulated information from the LEDs in its image pixels, along with the other scenery in camera view. The information gathered from the decoded VLC packets and the scene perception analysis from the receiver vehicle's camera are logically fused to generate relevant safety recommendations for the driver/vehicle.

The Augmented Vehicular Reality (AVR) work from Qiu et al. (7) takes an engineering approach by augmenting the scene beyond the driver's field-of-view to the driver's vision. This is achieved by communicating a compressed version of a camera image point-clouds (8) from the occluding vehicle and merging with those from the driver's vehicle camera. The key limitation of this approach is the requirement of a high-bandwidth link, such as a 60GHz (millimeter wave or mmWave) channel, between the transmitting and receiving vehicles to communicate the point-clouds, which are computationally expensive to generate. The system is highly error-prone as point-clouds are generated using a dense set of features from the scene and thus the robustness of the method relies on high feature extraction fidelity which is highly scene dependent. In addition, AVR provides only a visual augmentation of the occluded scene with no further information (e.g. warning) about the information perceived from the scene.

2.1.1 Novel Contributions

The key novelties of the proposed approach are: (i) intelligent information abstraction from scene analysis to communicate safety-critical data, (ii) dual usage of a camera for scene registration as well as an optical communication reception device, and (iii) achieving vehicle-to-vehicle (V2V) communication of safety information through a low data-rate link.

In summary, the contributions of this paper are:

1. Design of a novel architecture that uses DNN, computer vision and camera communication for achieving non-line-of-sight (NLOS) perception.

2. Implementation of a robust DNN model using transfer learning for vehicle brake-light detection and an annotated brake-light dataset (publicly available);

3. Design of a novel intelligent packet communication mechanism for NLOS perception that fuses camera scene perception with camera communication information;

4. Implementation of an trace based real-time functioning prototype of safety event identification under NLOS perception using off-the-shelf components;

5. Experimental evaluation of the prototype system in controlled and real-world settings.

The rest of the sections in the paper are organized as follows: Section 3.1.2 discusses related work, Section 2.1.3 presents an overview of the NLOS perception system architecture, Section 2.1.4 discusses scene perception module, Section 2.1.5 discusses the intelligent information mapping module, Section 2.1.6 discusses LED-Camera communication and the recommendation generation modules, Section 2.1.7 discusses prototype implementation, Section 4.1.6 discusses the evaluations and Section 4.1.7 concludes the paper.

2.1.2 Related Work

Vehicular Camera Communication. The fundamental idea of camera communication (9) is to use off-the-shelf camera devices as communication receivers for information that is encoded, as digital pulses, in light beams emitted from light emitting diodes (LED). The concept of camera communication has caught significant attention in the intelligent vehicular networking, transportation systems and autonomous driving communities (10; 11; 12). Cameras as receivers for optical communication in vehicles (13; 14; 15; 16) are attractive, as they can enable multiplexing information from multiple LED transmitters (or vehicles) and use of computer vision object tracking mechanisms. In (17; 18), the authors design a custom CMOS image sensor to detect LED transmitter efficiently by using a visual marker and also for optimized for high data rate reception (upto 54Mbps). In (19), the authors use the rolling-shutter phenomenon of CMOS image sensors to improve data rates and propose a new modulation scheme to address ambient noise and interference. The fundamental limitations of prior work in vehicular camera communications is that the design is either limited to short ranges, low data rates and cannot work under mobile conditions or that they are highly customized for better reception, which means the camera cannot be used for other applications. In addition, no prior work in camera communication has used the concept for realizing NLOS perception in vehicles.

Safety information communication using V2V and V2I. Recent works have proposed designs and also demonstrated prototypes of using visible light communication (VLC) for V2V and V2I, that use camera receivers (20; 21). While (20) uses the optical communication

link as a replacement for rear-view mirrors, (21) enables traffic light to vehicle communication upto 70m range using camera communication. The V2V VLC system designed in (22) uses OFDM techniques for noise mitigation in the photodiode receiver and has been tested on a real roadway delivering a range of 45 meters. Recent works (23; 24; 25) have also explored the use of traffic light LEDs to communicate traffic and vehicular positioning information, as proxies of safety information, to optical receivers on vehicles. The challenge with the use of these systems for NLOS perception is that they are dependent on specific infrastructure requirements (transmitter on traffic light only), do not address mobility (tracking with photodiodes is challenging), and that the systems do not account for the contextualizing the scene/safety events (VLC is the communication mode, however, generation and identification of dangerous events/safety warnings is key).

Vehicular scene perception and behavior detection. Over the past few years, computer vision based analysis has been investigated in vehicular networks to estimate vehicle's position, pose, distance and also to detect lane, traffic signals or pedestrians on the road (26; 27; 28; 29; 30). The insights and developments from these systems provide a basis for the proposed scene perception in our work, however, these existing works do not address the NLOS perception issue in its entirety.

2.1.3 System Architecture

We design a system for achieving NLOS perception in vehicles based on the fundamental idea of identifying safety critical events from visual scene perception, converting it into digital information and communicating it to other vehicles in vicinity. The system uses computer

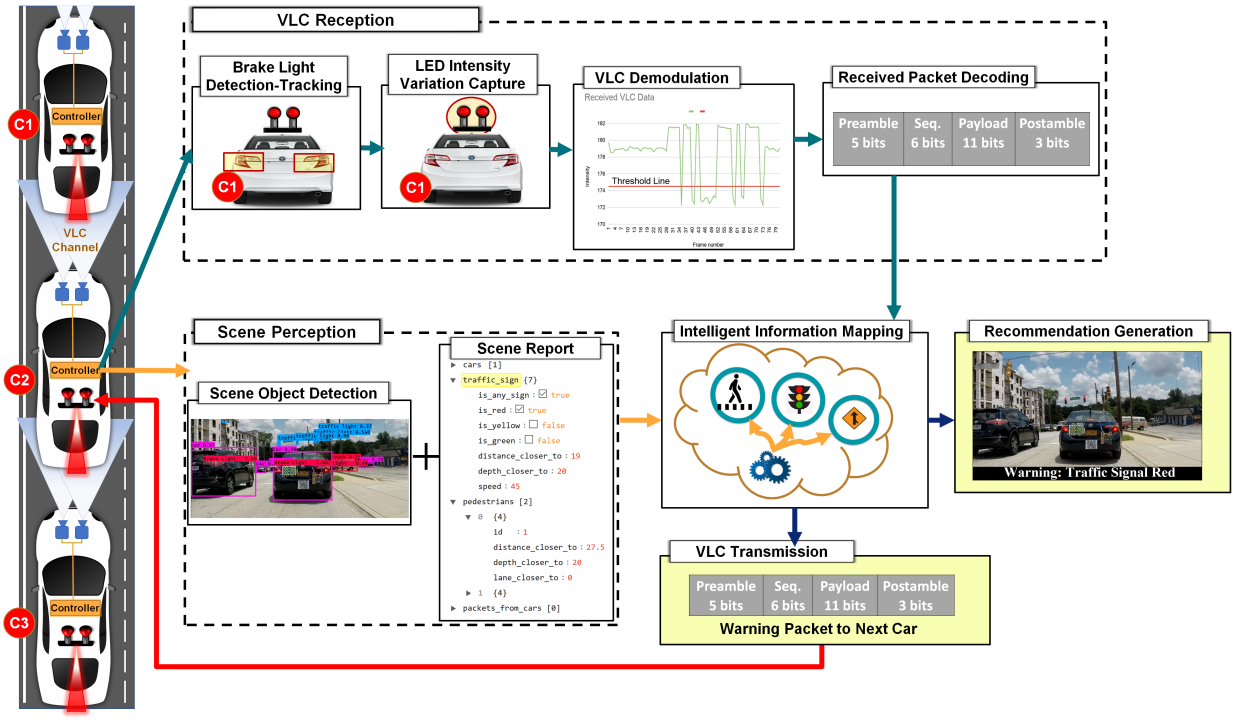


Figure 2.2: Illustration of the proposed NLOS perception system architecture. In this example, car C1 communicates to C2 informing of an event in the NLOS region ahead of C1, and car C2 communicates to C3 informing of an event in the NLOS region ahead of C2. In this example, the most critical event ahead of C2 is the status or behavior of C1, which is originally invisible to C3 due to occlusion by C2.

vision perception analysis to infer the safety critical event by analyzing the (camera) real-time footage. An information mapping unit maps the inference into warnings and digitally encodes into short data packets. The data packets which carry the safety warning along with a temporary unique ID (pertaining to the vehicle) is communicated to the immediately following vehicles on the road using LED-camera communication. The vehicle receiving these packets fuses the safety warning information along with the scene report from the perception analysis, to recommend an action to safety for the vehicle.

Our NLOS perception system design treats each vehicle as a transceiver; equipped with both, an LED (optical) transmitter that is used to communicate the packets, and a camera

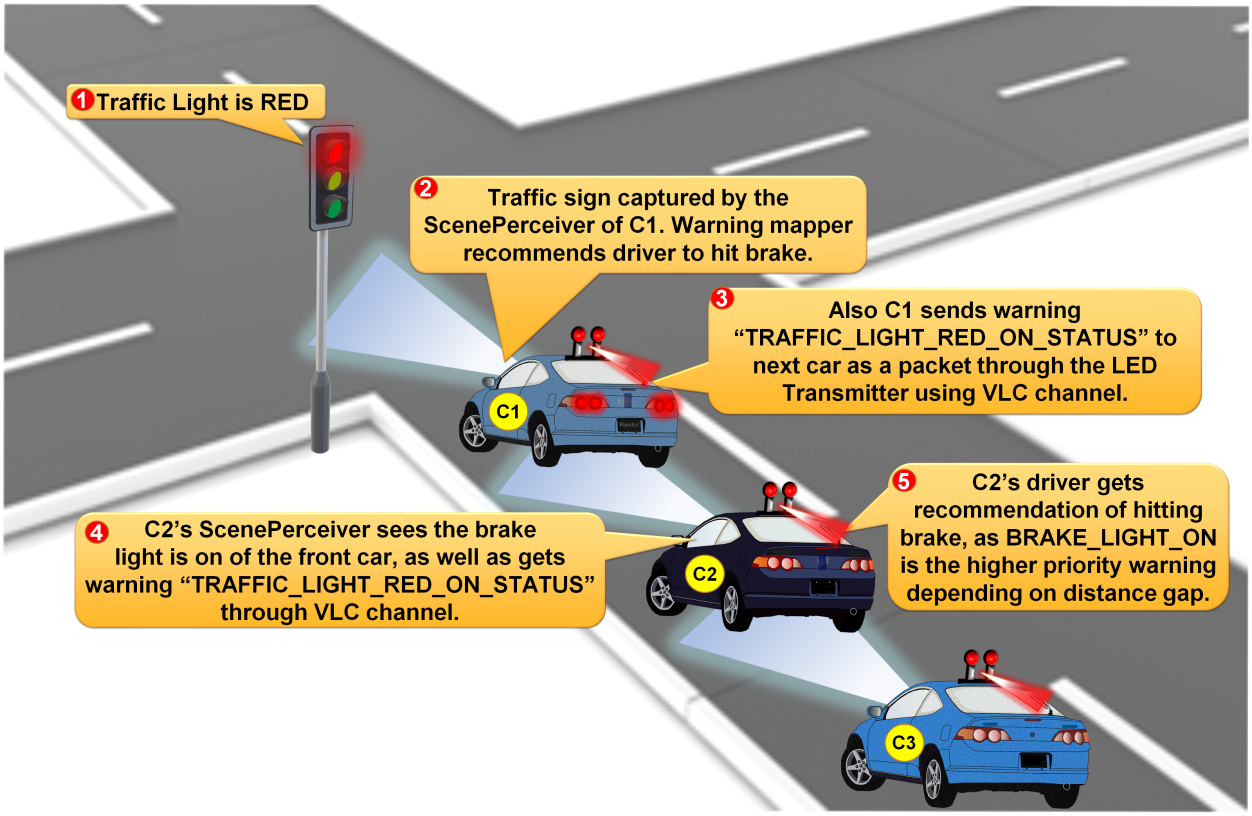


Figure 2.3: An example execution of our NLOS perception system. In this example, the TRAFFIC LIGHT IS RED event or warning is perceived by C2 with the help of C1, and the FRONT CAR'S BRAKE LIGHT IS ON (vehicle is slowing down/stopping) event is perceived by C3 with the help of C2.

receiver to decode packets and conduct scene analysis. This way, the information perceived by a vehicle can be communicated to the vehicle following behind, which can further fuse the collective information within its local context and communicate to its following vehicle, and so on. This transceiver based system design approach enables to build a pipeline for information transmission across a series of vehicles on the road, thus, creating opportunities for multi-hop and relay communications across vehicles.

The proposed design assumes that each vehicle is, or can be, equipped with a stereo-camera, LED emitter, and graphical processing unit (GPU) embedded controller. We posit

that this assumption is well justified considering the latest advancements in vehicles. Cameras, especially, stereo-cameras (also referred to as depth or 3D cameras as they can capture distance between the object and camera, along with the pixel intensities in x-y on the 2D image), are increasingly being integrated in vehicles to provide advanced driver assistance (ADAS), lane change monitoring, parking assistance, and for autonomous driving capabilities. Vehicles already equip three forms of LED emitters; headlight, tail lights and brake lights. Vehicle brake lights function like any other off-the-shelf LED, and thus can be used (in addition to illumination) as an intelligent visible light transmitter. Vehicles are also increasingly being equipped with more sensing and computing capabilities, and use of GPUs are becoming more common than ever in vehicular electronic control and computing units.

Our proposed NLOS perception system for vehicles includes four key components designed as modules with specific goals. These modules are further divided into sub-modules that have delineated functionalities:

Scene Perception. [Goals: Conduct visual scene perception to detect key markers on the road, vehicles, pedestrians, and localize the LED transmitter.] This module involves performing object detection and scene analysis. The object detector locates vehicles, pedestrians, and traffic lights. The scene analysis uses the object detection information to create a scene report summarizing the traffic light status, vehicle lane change status and pedestrian presence status.

Intelligent information mapping: [Goals: Convert the visual scene perception inferences into information that captures vital information about the event that is being occluded to

the driver.] This module hosts a warning mapper to compress the scene report by mapping the information into specific pre-defined road safety warnings. The warning is encoded into bits along with specific identity information bits padded into a packet using the packetization sub-module. Each packet contains one high priority warning which is decided based on a dynamic prioritization scheme by the warning prioritizer sub-module.

LED-Camera communication: [Goals: Transmit the information to the vehicle following behind, and Receive the information from the front vehicle.] This module integrates two sub-modules: transmission and reception. The transmission module converts or modulates the bits from the data packets into the HIGH/LOW or ON/OFF status of the LED emitter(s). The reception module hosts a LED emitter localization mechanism that performs brake light LED detection and uses that as a reference to locate any other LED emitters on the vehicle. The LED localization output gets updated into the scene report. Next, it conducts demodulation of bits encoded as LED ON/OFF status by processing the LED emitter localized regions on the camera image. The reception module also hosts functionality to decode information from the stream of demodulated bits.

Recommendation generation: [Goals: Generate appropriate recommendations to the driver/vehicle.] This module fuses the information decoded from the packet with the information from the intelligent information mapper module to generate an appropriate recommendation to the vehicle/driver.

We use the visual illustration and example in Fig 2.2 to describe how the bespoke modules function together in our proposed NLOS perception system. The example scenario in Fig. 2.2

considers cars C1, C2 and C3 driving on a single lane, one behind the other. The region beyond C1 is being occluded to C2, and the events related to C1 are being occluded to C3 (by C2). The goal of our system is to ensure C2 is informed of the events happening in the occluded region in front of C1, and C3 has to be informed about the events regarding C1. We will consider that all the three cars are equipped with our NLOS perception system, including the hardware components (stereo-camera, LED emitter and GPU computing unit).

Let us understand the NLOS perception system cycle from the perception of its execution in car C2: the **scene perception** module in C2 detects vehicles, road signs, traffic lights and pedestrians, and generates a scene report. The scene report is converted into a short packet by the **intelligent information mapping** module. This is done by mapping the scene report information into predefined warnings related to the event in front of the car; in this case, the event is the status/behavior of C1. The **LED-Camera communication receiver** module of C2 first detects the brake lights of the detected vehicles. In the case where the brake light is not used as the transmitter, the module uses the brake light location as a reference and performs a geometrical localization mechanism to locate the custom LED transmitter. The receiver, then, processes the pixelated region of the localized C1's LED emitter to decode the information transmitted from C1. The packet transmission using the LED is achieved by the **LED-Camera communication transmitter** module, where, the bits are modulated as intensity variations of the LED emitter using HIGH/LOW or ON/OFF keying (OOK), and transmitted at a specified transmission frequency (rate). The **recommendation generation** module in C2 intelligently combines or fuses the infor-

mation from its scene report with that from C1's packet, to generate an appropriate safety recommendation to C2. The scene report from C2 is further passed through its intelligent information mapping and LED-camera communication transmitter modules to communicate the information about C1's status to C3, which further executes the NLOS perception cycle to retrieve information and further relay the same. An example of the system's functionality for a RED traffic light warning use–case is illustrated in Fig. 2.3.

2.1.4 Scene Perception

The scene perception module creates a holistic understanding of the scene, viewed by the stereo-camera, by detecting key markers and analyzing activity on the road. In particular, this module focuses on detecting the traffic light, road lanes, and pedestrians. It uses the detection outputs and the distance estimate from the stereo-camera, to analyze specific states and activity, including, traffic light states (RED, YELLOW or GREEN), lane-change behavior of front vehicle, and proximity of the pedestrian to the vehicle. We describe the object detection and scene analysis functions of this module in detail in the rest of this section.

2.1.4.1 Object Detection

We leverage the state-of-the-art open-source implementations in computer vision object detection to detect vehicles, traffic lights and pedestrians. We choose the YOLOv3 (version 3) (31) for this task as it comes packaged with pre-trained DNN models (32; 33) that can help detect our target 'objects' on the road with high accuracy and in real-time. YOLOv3

DNN Model	**Processing speed [FPS]**	**Accuracy**
Mobile Net	3.57	95.7
Faster RCNN	1.2	96
YOLOv3	7.5	98
Tiny YOLO	15	96.2

Table 2.1: Test-bench comparison of existing DNN object detection models for mobile devices, on a personal computer with a NVIDIA GeForce GTX1060 GPU. The accuracy and processing speed (in frames per second) are based on our evaluation dataset size of 1000 images.

trained in COCO dataset (34) detects 80 classes. In this work we leverage the vehicular, pedestrian detection, and traffic lights classes.

Given that the YOLOv3 platform yields best accuracy, comparable to other high accuracy mobile DNN models (35; 36), we confirmed our choice for the object detection framework based on a test-bench evaluation of the computation time for the models. We represent the computation time as a function of the effective frames processed per second metric in Table 2.1 for MobileNet (37), Faster RCNN (38), YOLOv3 and Tiny YOLO (31). Note that Tiny YOLO is a lighter version of YOLOv3 aimed at computation resource constrained devices such as internet-of-things (IoT), smartphones, Arduino and Raspberry Pi devices, and trade-offs with the accuracy achievable by the full YOLOv3 model for specific type of objects. Overall, we choose YOLOV3 as we clearly observe that YOLOv3 has the best accuracy and with reasonable computation speed.

2.1.4.2 Scene Analysis

The scene analysis function aims to retrieve a perceptive understanding of the states and activities related to the specific objects detected in the road scene. This function creates a scene report of the holistic scene perception that is used by the intelligent information

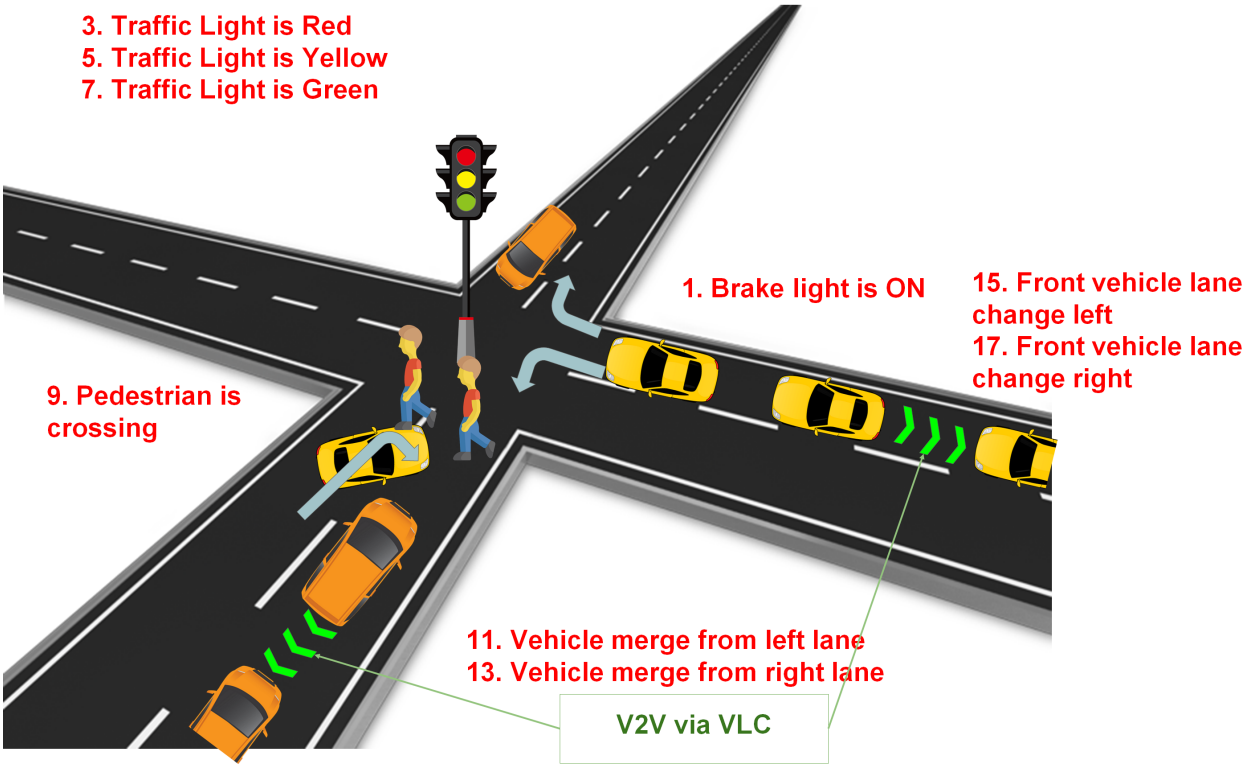


Figure 2.4: Depiction of the nine safety warning considered.

mapping module (section 2.1.5) to map with specific safety warnings that can be inferred from the scene.

Our system focuses on mapping safety warnings based on the inferences generated from nine road scene states and activities that are derived by the scene analyzer. As illustrated in Fig. 2.4, the nine safety warnings relate to traffic light status (# 3,5,7), pedestrian crossing/not crossing the road (# 9), front-vehicle brake light status (# 1), vehicle(s) in front view merges from left/right lanes (# 15,17), front-vehicle changing to left/right lane (# 11,13). We now discuss how these warnings are perceived by the scene analyzer.

Traffic Light Status (warnings # 3,5,7) The traffic light detection output from YOLOv3 provides the segmented localization of the lights on the image region. We use the masking

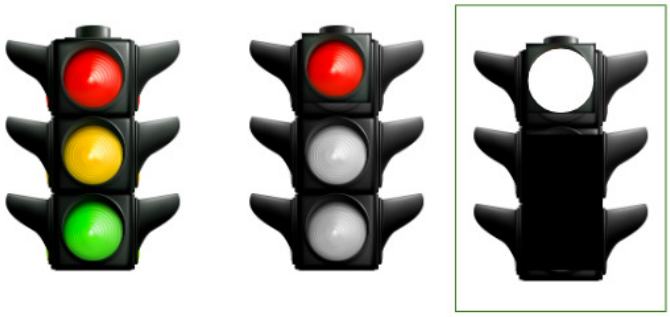


Figure 2.5: Traffic Light Analysis

process to detect the status of the traffic lights; if it is RED, YELLOW or GREEN. A mask, in computer vision, is a color range specification that is applied as any regular filter to the image. The mask flags and isolates all pixels in the selected region with color channel intensities that fall within the stipulated range. We use the standard RGB image color format in our system, and the masking process is provided a Lower mask ($[r_l, g_l, b_l]$) and an Upper mask ($[r_u, g_u, b_u]$). For example, in the Fig. 2.5, the left-most image shows the detected traffic light. The middle image shows the part of the image that falls in the range (in RED). We use the mask range of Upper Mask:(180,255,255) and Lower Mask:(175,50,20) for RED. The right-most image shows the final output where only the red part of the object is flagged and colored in white (pixel intensity as 255) and all other regions blackened out (pixel intensity as 0). Similarly, specific mask ranges are applied for YELLOW [Upper Mask:(25, 255, 255) and Lower Mask:(10, 100, 20)] and GREEN [Upper Mask:(70, 255, 255) and Lower Mask:(36, 25, 25)], respectively. The analyzer makes a decision for the region as a RED/YELLOW/GREEN based on the relative number of pixels in the flagged-as-white region.

To determine the mask thresholds, we conducted empirical trial and error experimenta-

tion for each of the colors. In our experimentation we collected images of a traffic light in broad daylight at distances ranging from 2m to 50m. We selected a mask threshold for each color based on the constraint that the color must be detectable up to 50m distance. We tested the mask thresholds in a diverse set of real world camera images of vehicle driving on roadways, collected from our experiments and from public vehicle driving datasets (39). Based on our empirical analysis, we set the threshold for the number of pixels as 1 %; that is, if there are more than 1% whitened pixels of the total number of pixels in the detected region, the analyzer marks the status of the traffic light as the appropriate color (RED in this example).

Pedestrian Crossing Status (warnings # 9) The basic object detector localizes the pedestrians on the image. The scene analyzer refers to the position of the pedestrian in the image and uses the distance (depth) to the pedestrian value from the stereo-camera API to determine if the pedestrian is crossing the road in front of the vehicle and how far is the pedestrian from the vehicle. The analyzer uses the lane analysis output (discussed next) to determine the lane in which the pedestrian is detected.

Vehicle Behavior from Lane Analysis (warnings # 11,13,15,17) The analyzer estimates the lane change behavior of the vehicles in front. It identifies if any vehicle is merging from a left or right lane, and if the front-vehicle in the same lane is trying to change to the left or right lane. To estimate such behavior, the analyzer first conducts a lane analysis, in which it demarcates the boundaries that divide the current driving lane from the LEFT and RIGHT lanes. The lane analysis is a geometrical calculation of the regions on the image that

can approximately divide one lane from another. As illustrated in Fig. 2.6, the lane analyzer uses an isosceles triangular region centered on the front-vehicle as the region of interest that marks the current driving lane (Lane-0). If a vehicle is detected to the (reader's view) outside left of the region, then it is marked as LEFT lane (Lane 1). The right lane (Lane -1) follows an identical logic for the right side view. The scene analyzer, uses the lane analysis information from each frame of the video stream over a stipulated time window (we use 1 sec in our prototype system) to determine the series of lane positions in these frames to mark whether the front-vehicle changed its lane and if any other vehicle merged or is trying to merge with the driving lane. The behavior of the vehicle in each session is determined using a set of 3 frames selected from 100 frames sampled in 1 sec (every 33rd frame), compared with the history over the last 4 seconds. Here, the decisions are not based on analysing a single frame using the triangle threshold. This way even if the vehicle is marked as "out of the triangle" we do take into consideration the trajectory of the vehicle over that last few seconds. Such a trajectory tracking and vehicle behavior detection was already implemented in our prior work (40).

In the event that the front car is on a curved path, it is, geometrically, very challenging to address this case. This is a practical challenge, even for the state-of-the-art (semi) autonomous vehicles. We note that we do not claim any novelty in the lane change analysis aspect of our design. We identify that it is possible to integrate the road map information from third parties, e.g. Google Maps, and have the vehicle be informed that there are curved roads in the scene sampled at that time instance. We reserve such considerations for

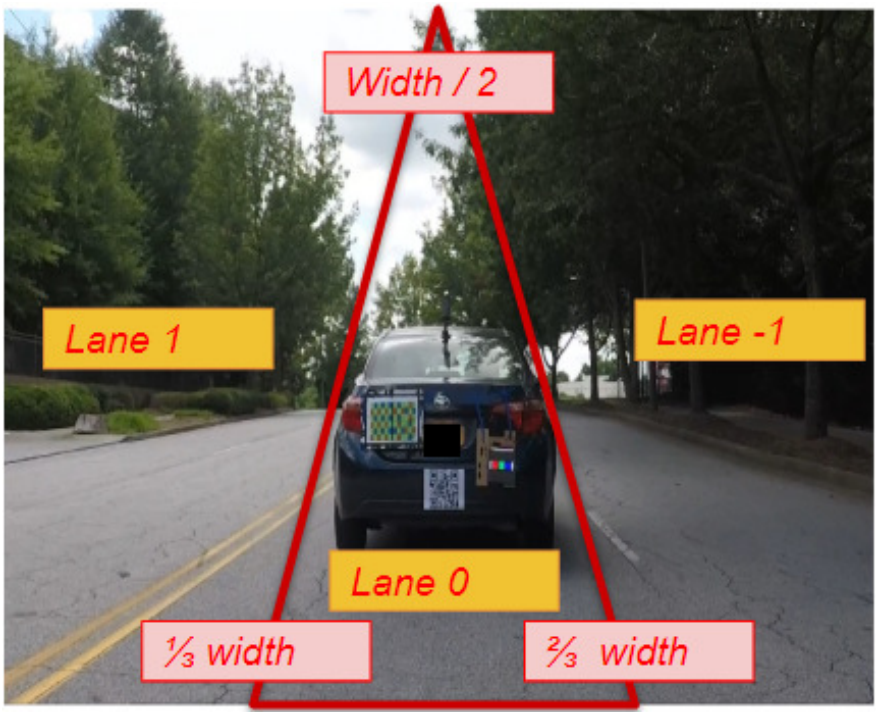


Figure 2.6: Illustration of lane analysis. The vertices of the isosceles triangle is chosen relative to the width (number of columns) of the image.

future work. The outputs from the scene perception module are collectively summarized in a compact JSON (41) format, referred to as a scene report, as shown in Fig. 2.7.

2.1.5 Intelligent Information Mapping

The goal of the intelligent information mapping module is to create a concise representation of the exhaustive set of information listed in the scene report. This process is analogous to compressing the rich set of data in the scene report representing the detected objects and inferred activity from the scene. The rationale for compressing the scene report is to help create a compact representation of the scene report in a holistic form that can be communicated or relayed to other vehicles in vicinity. The intelligent information mapping module performs this compression in the form of a contextual mapping of the scene report

```
{
  "cars": [
    {
      "id": 1,
      "is_any_car": false,
      "relative_speed": 5,
      "distance_closer_to": 5,
      "depth_closer_to": 7,
      "is_brake_light_on": true,
      "is_hazard_light_on": false,
      "lane_closer_to": 0,
      "is_leaving_lane_left": false,
      "is_leaving_lane_right": false,
      "is_merging_lane_from_right": false,
      "is_merging_lane_from_left": false
    }
  ],
  "traffic_sign": {
    "is_any_sign": true,
    "is_red": true,
    "is_yellow": false,
    "is_green": false,
    "distance_closer_to": 19,
    "depth_closer_to": 20,
    "speed": 45
  },
  "pedestrians": [
    {
      "id": 1,
      "distance_closer_to": 27.5,
      "depth_closer_to": 20,
      "lane_closer_to": 0
    },
    {
      "id": 2,
      "distance_closer_to": 27.5,
      "depth_closer_to": 20,
      "lane_closer_to": 0
    }
  ],
  "packets_from_cars": []
}
```


Figure 2.7: Scene report in JSON format

information to safety warnings.

The safety warnings as defined in our system are a set of contextual representations of the actual scene perceived and reported in the scene report. The notion of these warnings

Warning ID	Safety Warning
1 (2)	BRAKE_LIGHT_ON
3 (4)	TRAFFIC_LIGHT_RED_ON
5 (6)	TRAFFIC_LIGHT_YELLOW_ON
7 (8)	TRAFFIC_LIGHT_GREEN_ON
9 (10)	PEDESTRIAN_IS_CROSSING
11 (12)	VEHICLE_MERGE_FROM_LEFT_LANE
13 (14)	VEHICLE_MERGE_FROM_RIGHT_LANE
15 (16)	FRONT_VEHICLE_LANE_CHANGE_LEFT
17 (18)	FRONT_VEHICLE_LANE_CHANGE_RIGHT

Table 2.2: List and encoded IDs of safety warnings.

is to help communicate the most important contextual meaning of the occluded scene to the behind vehicles, so that those drivers (vehicles) can take appropriate safety action in a proactive manner. In the below paragraphs we setup ground of selecting a single most important warning combining line-of-sight and non-line-of-sight information.

For a driving agent (human/machine) the notion of safe driving is to understand a sequence of road scenes and respond successfully to the most critical event associated to those scenes (42). For example, upon identifying as sudden brake of the front vehicle adjusting speed in response, or changing lane in response to the irrationally merging vehicle, stopping the vehicle in response to a sudden Yellow light changing to Red. Even if, multiple critical event arises at the same time, for example, a typical highway multi-vehicle collision scenario, speed truck crashing multiple vehicles causing swerving car from side lane and front car sudden stop, because of the vehicular dynamics the ultimate single action could be either escaping the collision by moving the vehicle to the safer part of the road (LANE CHANGE), or bringing the vehicle to immobile state (STOP) immediately to lessen the crash impact. This narration helps conclude that the leader vehicle must transmit a single and most critical warning to the follower vehicles.

In our system, we define 9 safety warnings, that are mapped to information in the scene report. As listed in Table 2.2, we index each warning using positive integers starting from 1, and use 2 successive integers as indices for each warning, which we refer to as warning ID. We use gray-encoding (43) of each index as the binary representation of the warning. The notion of using multiple indices and gray-encoding is to ensure that each warning code has a consistent hamming distance (43) (number of differing bits) of 2. We represent each warning using 5-bits, which allows for a total of 16 warnings. Based on the focal object detection and scene activity in our system, our current system implementation encodes 9 warnings – an expansion to 18 warnings is straight-forward with our prototype. The communication strategy is to inform other vehicles in vicinity of the particular warning by encompassing the ID in the communicated data packet. The warning ID code is communicated if the logical status of the warning based on the scene report is TRUE. The default value of a warning ID is set to all-zeros if the corresponding status is FALSE.

The key rationale for the mapping the scene activity to safety warnings is based on the overall goal of the system – to ensure the safety of the driver (vehicle) when there is occlusion (NLOS) on the drive path. It is definitely useful to translate the scene report into an exact digital representation or data (bits) and communicate to the vehicles behind and in vicinity. However, this approach is challenging to scale as the communication of the scene report, as a data file, will require a highly reliable and high-bandwidth link between the vehicles. In addition, the data rate requirement will change depending on the size of the scene report as the density of data in the scene report varies with the richness of the scene.

We posit that the communication of the scene report, as a file, is however, a middle-ground solution between communicating the scene images/footage/visual features/point-cloud and as a highly compressed format using our proposed mapping. In this paper we largely focus on communicating using the mapping approach, however, our fundamental framework is adaptable to communicating the entire scene report as a file using the proposed LED-camera communication channel.

2.1.6 LED-Camera Communication and Recommendation Generation

The LED-Camera communication module is responsible for reliably communicating the information from one vehicle to another. The idea is to communicate the highly compressed and contextualized scene report information in the form of warnings. We posit to use camera communication (9) where information is communicated using the light beams emitted by an LED transmitter (modulated and encoded) and received (demodulated and decoded) by a camera. The rationale for using LED-Camera communication as a part of the NLOS perception solution, is 3-fold:

1. Dual use of the camera: Our design enables to reuse the camera, originally serving as a scene capture device for perceiving the scene, as a communication receiver for the same scene perception information.

2. Use existing infrastructure: Our design enables reuse of existing infrastructure such as LEDs on brake-lights and tail-lights in vehicles and cameras integrated in vehicles – such as for advanced driving assistance (ADAS), parking assistance, lane change

assistance, and perception for automated driving.

3. Works even with low data rates: Our design makes the system work even if the communication data rates are low. Cameras are limited in sampling rates (frame rates) compared to possibilities of radio receivers or even photodetectors. However, this does not create a bottleneck in performance of our system as we consider the very commonly available sampling rate. We leverage the fact that 100 FPS cameras are becoming more common place (smartphones), and hence design the camera communication system considering 100 FPS as the minimum sampling rate requirement for the performance and features demonstrated by our system.

The communication module functions are two-fold: (i) LED transmitter and (ii) Camera receiver. The LED transmitter converts the mapped warnings into signals that can be communicated using the optical channel. The camera receiver converts the optical signals from the LED which it detects in the form of pixel intensities registered on the camera sampled images. Upon successful reception, the receiver retrieves the communicated safety warning. This warning message is then mapped into an appropriate recommendation for action to the driver or the vehicle, by the recommendation generation module. We describe these modules in detail in the rest of this section.

2.1.6.1 LED Transmitter

The safety warning generated and encoded as 5 bits by the intelligent information mapping module is packed into a compact data packet. Based on the constraint (limit) in the camera

Field	Attribute	Size (bits)	Notes	Sample values
1	Preamble	5	5-bit Barker code	11101
2	Packet ID	6	Sequentially increasing integer	101000
3	Distance	5	Distance to the nearest threat	01101
4	Warning ID	5	information mapping module output	00101
5	Parity	1	Error checking	1
6	Postamble	3	3-bit Barker code	110

Table 2.3: Data packet structure

sampling rate of 100 FPS, by following Nyquist criterion (43), we set the target transmission rate at 50Hz. We therefore choose a packet size of 25 bits, with a notion to at least transmit 2 packets/second.

It is possible that each scene report can have multiple safety warnings being generated. Our system decides which safety warning has to be transmitted based on a priority scheme, where the warning with highest priority (among all applicable warnings) is communicated. The warning prioritizing method has been discussed in section 2.1.5.

Packet structure. We structure a data packet in our NLOS perception system as shown in Table 2.3. In addition to the warning ID of the safety warning to be communicated, each packet also carries the distance information. The distance value, in meters, noted in each packet (as a binary number) represents the distance to the most imminent 'threat' to the vehicle, which translates to the distance of the nearest object on the drive path as noted in the scene report. Each data packet also consists of a preamble and a postamble that are used for synchronizing the packet starting and ending points. We choose barker sequences (44) in this regard as they are known to have high auto-correlation property, and thus it is possible to identify the start and end of each packet through a simple sequence correlation on the

receiver. We choose a 1 bit marker as a parity bit (43) for error check. We index each packet with a packet ID, which represents the packet sequence number transmitted in each transmission session and which resets to 0 at the start of every new transmission session.

Packet transmission scheme. We use asynchronous packet communication in our system. The packets are transmitted periodically at the stipulated rate (2 packets/sec) over each transmission session of S seconds, thus creating a packet information redundancy of $2S$. We use this repetition (redundancy) of packets to ensure reliable reception at the camera receiver. In our current system prototype we have achieved reliable system performance for $S = 1$. This is based on our empirical evaluations based on which we inferred that having at least 2 duplicates of a packet at the receiver helps achieve a performance that can be practically accepted. As we will discuss in section 2.1.8.3, a 1 second session duration entitles to the transmission and propagation times in our system. The total output response time (transmission + propagation + processing) for our system is 1.3248 seconds (illustrated in Fig. 2.12). Considering the asynchronous modality for reception and processing, this processing time will not impact the sampling of the camera frames and caching of the packets in subsequent intervals.

Modulation. The communication of the bits in the packet is implemented by modulating the light emitted from the LED. We base our modulation mechanism on ON-OFF-Keying (OOK) (45), traditionally used in optical wireless communication systems, where bit 1 is mapped as ON state of the LED and bit 0 mapped as OFF state of the LED. The fundamental challenge with OOK is the flickering phenomenon, which is a visual artefact where the light

emitter intensity seems to waver randomly or flicker to the human eye. The flicker effect is inversely related to the LED modulation frequency. Psycho-visual studies (46) have noted that light emitter frequencies above 100Hz are not easily noticeable by human eye and become insignificant as the frequency increases. At 50Hz (our choice of LED modulation rate) the flicker is highly prominent. This is due to the large intensity variations between the LED's ON (high) and OFF (low) states. To address the flicker issue, we map the ON and OFF states of the LED to pseudo-high and pseudo-low states, respectively. Considering the median value between the ON and OFF LED intensities as the baseline idle, we map bit 1 as idle $+\frac{\Delta}{2}$ and bit 0 as idle $-\frac{\Delta}{2}$. This way, the dynamic intensity range is Δ, and by choosing a small value of Δ the flicker may not be perceivable, even at the 50Hz frequency. A $\Delta = 40$ is chosen in our current system prototype based on a set of empirical evaluations, which we present in detail in section 4.1.6.

This Δ modulation helps in maintaining an effective average intensity throughout the modulation sessions. The modulation is on top of the baseline switch-ON signal of the LED and thus not perceived through human eyes. The visible light camera communication receiver detects the embedded HIGHs and LOWs and maps it correspondingly to 1s and 0s. In the rest of this paper, we refer to this modified version of OOK as Δ-shift-keying (DSK) modulation, and the baseline as LED idle state intensity – the idle state corresponds to the default (switched ON) status of the LED when it is not communicating any packet.

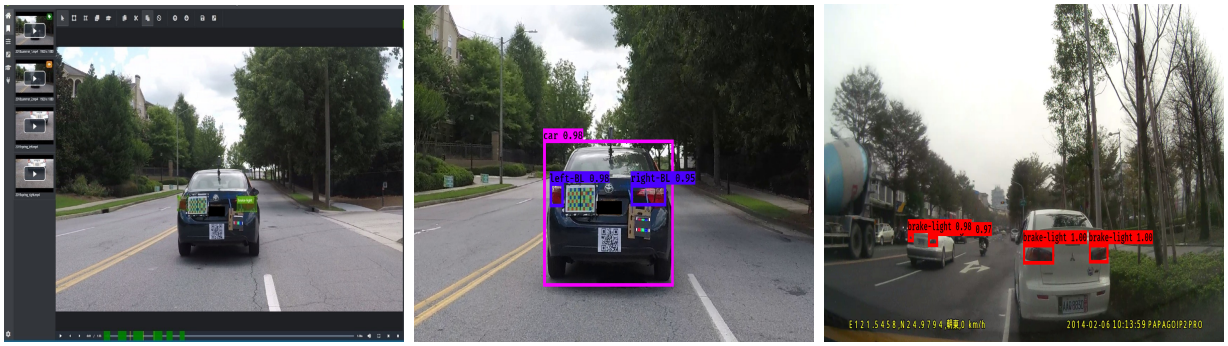


Figure 2.8: In series order from left, (i) Microsoft VoTT used to annotate the brakelights, (ii) Sample output from customized YOLOv3 model for brakelight for Atlanta images (the vehicle detection is from baseline YOLOv3), (iii) Sample output from customized YOLOv3 model for brakelight for Taiwan images.

2.1.6.2 Camera Receiver

The packets transmitted by the LED transmitter from the front-vehicle are received and decoded by the camera receiver module embedded in the scene perceiving stereo-camera. The data samples at the receiver are each image frame sampled by the camera. The reception module functions entail localizing the LED emitter on the camera images (at which pixels is the light from LED registered?), demodulating the registered LED intensities to bits (does the LED registered intensity translate to a 1 or a 0), and decoding the packet and hence the safety-warning (is this a data packet and what is the information encoded in the same?).

The LED emitter localization on the camera image frame requires detecting the LED. Object detectors, including the state-of-the-art YOLOv3 framework we chose, are not traditionally trained for detecting LED as these can be regarded as custom defined objects. Keeping the fundamental goal of our system to be applicable to any road vehicular setting, we address this problem through new DNN based approach that is independent of the type of LED emitter being chosen. The idea is to detect the brakelights of each vehicle in the camera

view and use the relative position of any LED transmitter (with respect to the brakelight) for localization. We choose the brakelight as it is a standard 'object' in each vehicle.

Brake Light DNN Model. We custom-trained YOLOv3 CNN deep learning model with our brakelight annotated vehicle image dataset (3245 images). We choose the transfer learning approach, due to the non-availability of any reliable open-source brakelight object detector and dataset. We created the dataset by manually annotating every LEFT and RIGHT brakelight of each vehicle visible in the camera sampled frame. The annotations were performed using the Microsoft Visual Object Tagging Tool (47). The camera images were sampled from a series of video footages captured from a stereo camera fit onto the dashboard of a car and driving around Atlanta (Georgia, USA) roads. We also appended the vehicle image dataset from the Taiwan research group (39) with the brakelight annotations. The annotations and sample outputs from our model validation are shown in Fig. 2.8. In practice, the LED emitter may be embedded into the brakelight or taillights of the vehicle or be a custom LED emitter placed on the vehicle. In any case, we assume that the placement of the LED transmitter, relative to the brakelights of a vehicle, is known to the receiver. The rationale for this assumption is that the geometry of the transmitters and receivers can be set as a default or standard in the NLOS system configuration settings which are agreed upon by different vehicle manufacturers who integrate our proposed system. By reliably tracking the left and right brakelights, the camera receiver localizes any LED emitter placement by using the geometrical relationship between the brakelight positions and mapping the same to camera perspective projection theory (48).

Demodulation. Tracking the LED emitters helps identify the pixels that carry the modulated bits (using DSK). The receiver considers a 10 x 10 pixel region centered on the centroid of the tracked LED region for demodulation. The average pixel intensity of the localized LED region is determined and the 1s and 0s are demodulated based on an adaptive thresholding based strategy. In this approach, the receiver first records the HIGHEST (H) and LOWEST (L) value of the received intensities over a 1 sec time-window. We choose 1 sec as the receiver samples at least 100 image frames in this duration to allow for registering at least 2 packet duplicates. If $\delta_r = 0.5(H - L)$, is the median of the dynamic range of the received LED intensities in the pixel domain, then the demodulation uses $\delta_r + \epsilon(\delta_r)$ as the threshold for demodulating the 1 (higher value) and 0 (lower value). In an ideal scenario, the LED signals are clean and registered without any blur or other motion based artefacts, and an $\epsilon = 0$ will work. However, in reality, we observed through extensive trial and error in various situations (indoor dark, indoor ambient lights, outdoor overcast, outdoor sunlight), an $\epsilon = 0.25$ provides good demodulation accuracy (low bit error rates). We note that any threshold based reception is susceptible to new artefacts, and the methodology can be further improved using adaptive signal processing approaches. However, in this paper, we posit to demonstrate the feasibility of the system even in real settings and reserve the upgrade and finer optimizations for future work. The demodulated bits are packetized and correspondingly mapped to the warning ID.

2.1.6.3 Recommendation Generation

This module maps the warning ID decoded by the camera receiver with the appropriate safety-warning registered in the standardized safety warning dictionary. In tandem, this module also constantly registers every (updated) information about the scene perception from the information mapping module. The module makes a decision on the recommendation to safety for the driver (or driving vehicle) based on these two levels of information.

Warning ID	**Recommendations**
1 (2)	DECELERATE
3,5 (4,6)	STOP
7 (8)	GO
9 (10)	STOP/PEDESTRIAN
11 (12)	MERGE ATTENTION LEFT
13 (14)	MERGE ATTENTION RIGHT
15 (16)	DEPART LANE ATTENTION LEFT
17 (18)	DEPART LANE ATTENTION RIGHT

Table 2.4: Recommended action for each safety warning.

In our prototype system, the decision making is based on which safety warning, between the one decoded from the received packet and the one from the scene report, carries the highest priority. Here, we use the same priority scheme used in the packet generation process at the transmitter. The safety warning that has been selected is mapped to an appropriate action to safety for the driver or the driving vehicle. Table 2.4 lists the mapping used in our system prototype, however, this can be easily reconfigured based on different requirements of users and situations. The recommendation action to safety could be an audio, audio-visual, visual or even an automated feedback to the driver or the vehicle's control.

The prioritization of warnings at each instance (session) is based on the urgency of the situation. We qualify the urgency of the situation by comparing how probable each event is

to happen in reality and quantify by comparing the distance of that event under question, from the observer car. If the event is within a safe-distance threshold (predefined as 15m in our prototype), it gets high priority. For example, a pedestrian crossing the road will be given the highest priority compared to a traffic signal light that is farther away. On the other hand, if the pedestrian is farther than the predefined safe distance threshold then the immediate occurring, for example, the traffic light event gets higher priority. If both events are within the safe distance threshold, then the recommendation to be taken to lead to safety will be compared. In this case, the recommendation (as in our prototype) will be to STOP the vehicle. Hence, regardless of the priority the vehicle is directed to a safe decision.

In our prototype, specifically, we incorporated the warnings for traffic lights, vehicles, and humans on the road. However, adding other warnings (e.g. a random object or traffic cone) into the warning vocabulary, including the decision making process, is straightforward as it the scene perception module can be independently updated. If the detected class on the roadway is non–vehicle and non–human then the vehicle must prioritize slowing or stopping or diverting, considering that the car can potentially collide with this object. The system keeps a track of the distance between the vehicle and the nearest detected object, and marks to stay clear of this obstacle based on the safe-distance to clear parameter.

2.1.7 System Prototype Implementation

Hardware Implementation. The hardware configuration of the system includes three key parts: (i) camera, (ii) processing unit, and (iii) Red LED transmitter unit. The LED (49) is circular in shape and its dimension is 10.8 x 8.3 x 3.1 inches. The LED is powered using a

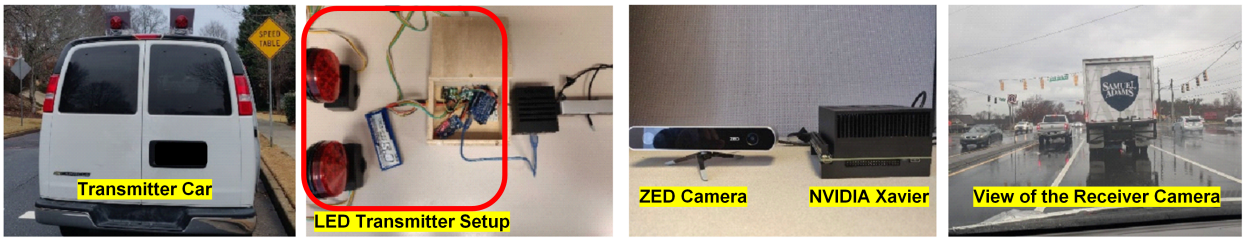


Figure 2.9: From left, transmitter car that carries the LED Lamps, hardware setup of the LED transmitter, the ZED camera receiver with NVIDIA Xavier, and one image snapshot of the dashboard mounted ZED receiver camera.

11.1 Volt 6 cell Lithium Polymer (LiPo) battery. We use a ZED stereo camera (50) for the camera unit. We operate the ZED camera at its maximum frame rate of 100 FPS, achievable at VGA resolution (640 x 480 pixels). The ZED camera provides two images (left-camera view and right-camera view) at this resolution, and also a depth image where the distance (depth) from the camera to the scene in each pixel is recorded. We use a NVIDIA Jetson Xavier AGX (51) development board as the processing unit in our system. The need for using a GPU board is to realize the computer vision and DNN processing in real-time and also enable the potential for online learning (training). We use a standard off-the-shelf vehicle brakelight/taillight LED stop-lamps (52) for the LED emitter which is controlled using an Arduino Uno microcontroller. We use 2 LEDs in our prototype system, just to provide flexibility in potentially multiplexing data streams from multiple LEDs. To enable regulated power supply from the LiPo to the stop-lamps we use MOSFET switches, and we wire the two MOSFETS in parallel to maintain the same input voltage across the stop-lamps. We use a buck converter to regulate the input voltage to the Arduino to 5 V (recommended voltage for standard Arduino Uno operation). Communication to the Arduino Uno was established through a serial-to-USB connection with the Xavier AGX. The transmitter setup is placed in

a wooden box, that was hand cut by us, to house the wiring and components, for portability. In addition, a XT30 connector is used to facilitate easy replacement of the LiPo battery pack.

Software Implementation. We use Python (ver. 3) as the standard programming framework in our system. We control/program the ZED camera using its available Python API. We have implemented object detection on Keras and using Tiny YOLO (on YOLOV3 (31) framework) framework for the DNN model. The Keras framework simplifies the integration of Tensorflow (53) with python libraries and to run YOLOV3 effectively on the GPU. We use OpenCV (54) and Pillow Python libraries for image processing and computer vision processes. The LED transmitter is controlled using an Arduino Uno which is serially connected to the Xavier AGX GPU processing unit. The control of the Arduino and the transmission codes are executed using serial function calls to the Arduino directly from the system's main (Python script) function in the GPU. The calls essentially initiate different C scripts on the Arduino based on the required functionality of the transmitter at each time instance. Typically, the transmitter is triggered for execution as soon as the system switches ON and the Arduino executes an appropriate script depending on whether the main function calls for TRANSMIT or IDLE. The system initiates processing as soon as an image is captures and conducts the relevant processing of each modules. The system uses a total of 3 images (separated by 30 image frames each) for the scene perception and associated mapping modules, and uses 100 frames for camera communication receiver module.

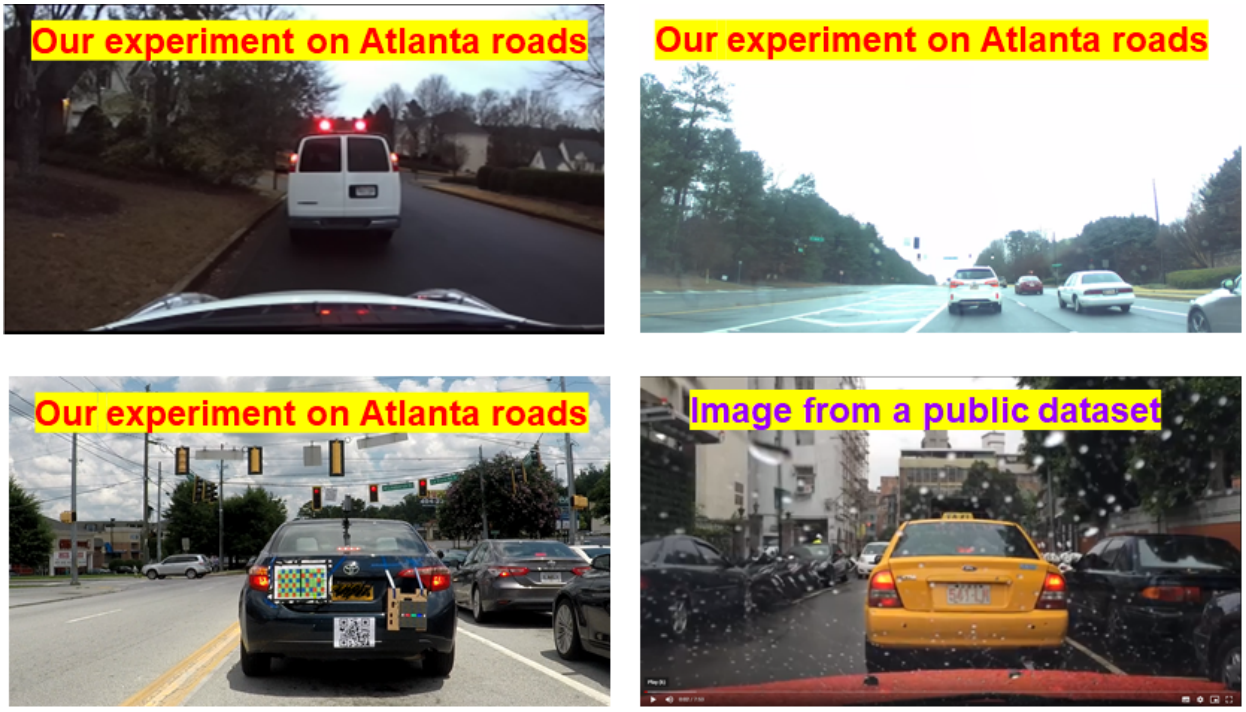


Figure 2.10: Sample image frames used in the system. Our collected data in the Atlanta roads are used to evaluate the proposed system. The public data set was used to train the brake light detection model for LED emitter localization.

Warnings	**Correct Detection**	**Wrong Detection**	**No Detection**
Pedestrian	99	0	1
Brake-light	90	0	10
Merge Left	98	1	1
Merge Right	90	2	8
Leave Left	89	5	6
Leave Right	90	6	4
Traffic Green	80	10	10
Traffic Red	87	8	5
Traffic Yellow	84	10	6

Table 2.5: Scene Perception Accuracy [in %], for each of the nine warnings in our NLOS scene perception system. The number of samples in the evaluation for each warning from left to right, are: {78, 129, 121, 44, 13, 14, 78, 85}. The variable sample numbers are due to the fact that a real-world road scene footage is highly scenario dependent and any of the nine warning situations may arise with variable distributions.

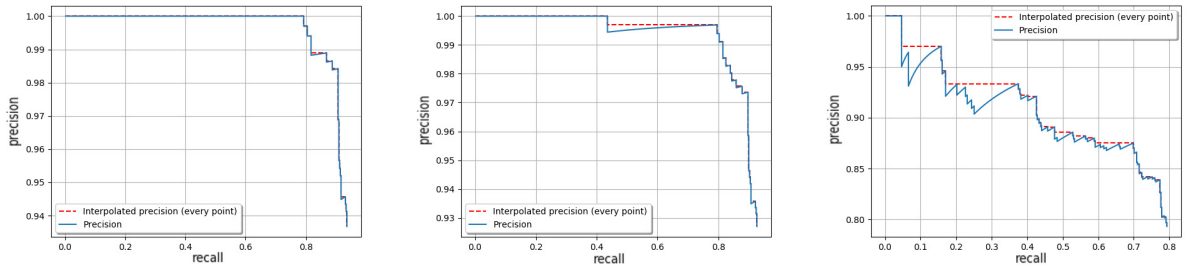


Figure 2.11: Brake light detection: Precision vs. Recall graphs for IoU = 0.50 (left), 0.75 (middle) and 0.95 (right).

2.1.8 Evaluation

We evaluate the performance of our NLOS perception system design through experiments. We evaluate the system based on the performance of, (i) scene perception, (ii) LED-camera communication and (iii) real world trace-based safety recommendation. The key evaluation metrics include, accuracy, packet error rate (PER), mapping error, and computation time. For the experiments, we setup the receiver on a car, which involved mounting the ZED camera on the dashboard of the car and running it at 100 FPS and be controlled by a Jetson Xavier board. We set up a LED transmitter system (see Figure 4.4) on another car by mounting two LED stop-lamps (52) on to the roof of the car using magnetic mounts. Since we did not have direct access to the brake light of the car used for experiments, we purchased and used actual off-the-shelf vehicle brake-lights, to emulate the functionality of real-life brake lights as we did not have permission to physically open the brake lights of the cars we used for experiments. The evaluation considers these LED brake lights ONLY for the performance analysis. We do note that the front-vehicle's actual brake lights will also be detected through our brake-light deep learning model. Once the actual brake lights are tracked we then use geometry to track the LED lights we placed on the vehicle. We

assume that we know the positioning of the LEDs on the vehicle. We drove the receiver car around Atlanta's city and sub-urban highway roads, and effectively collected about 4 hours of footage. Figure 2.10 presents a sample set of image frames from the recorded footage and from the public dataset. We had the controller process these videos and generate warnings in real time and communicate that using our LED brakelights. The experiments were conducted in a static setting and when the controller and the brake lights were placed under movements – placing it on a car.

2.1.8.1 Scene Perception

We evaluate the efficacy of the scene perception module based on the accuracy of detecting each of the nine warnings (refer Table 2.2). We computed the detection accuracy by analyzing the image frames from the previously mentioned real world footage from Atlanta roads. The images that contained the specific situation corresponding to the warning, were manually annotated with the warning ID as the label. Each image frame may have one or more warning ID, and hence each label was generated in a comma separated value (.csv) file format with the columns denoting a YES (NO) for the presence (absence) of the warning.

Table 2.5 shows the detection accuracy for the nine warnings, denoting the percentage of correct detection, wrong detection and detection failures (Not detected). Overall, we observe that our approaches for detecting each warning in our scene perception module are effective, with an average detection accuracy of 90%. We observe that there are certain cases of detection failures for all warning cases except pedestrian. Based on studying these failures we observed that the failures are primarily due to the failures in detecting the object by the

IoU (%)	50	55	60	65	70
AP (%)	93.40	93.40	93.29	93.29	92.76
IoU (%)	75	80	85	90	95
AP (%)	92.22	91.68	90.62	89.49	72.40

Table 2.6: Brake light detection: Average Precision (AP)

baseline YOLOv3 model. This failure/error percolates into the overall warning situation perception accuracy. We also observe that the traffic light cases suffer from detection errors, which are mainly caused by false-positives. Based on our manual examination of each of these erroneous cases, we observed that the errors were caused by confusion of the colors when the image was blurry or from a long distance, and when the traffic lamp was not fully detected.

2.1.8.2 LED-Camera Communication

We evaluate the efficacy of the receiver module based on the accuracy of localizing the LED emitter and the error rates of the data packet reception (PER) using the camera communication link. Our LED localization method depends on the geometrical mapping of the emitter location on the image using the detected vehicle brake lights' positions. In this regard, we first determine the detection accuracy for the actual brake lights of the vehicle. We then evaluate the data reception fidelity by measuring the PER in the LED-camera communication link. The LED localization errors are captured in the PER, as errors in finding the actual LED emitter locations in the image will lead to erroneous demodulation, and thus bit errors, leading to erroneous packets.

LED Emitter (Brake light) Detection: We evaluate the efficacy of our DNN brake light detection model through the Mean Average Precision (mAP) metric. Of the dataset of 3245

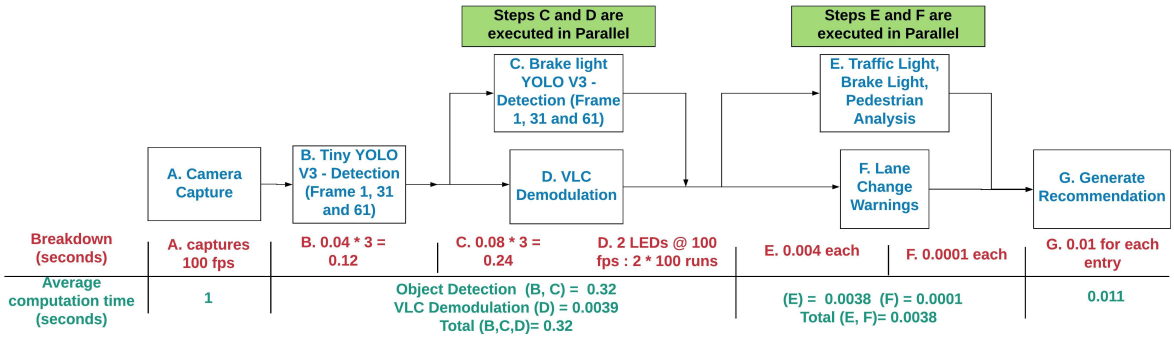


Figure 2.12: Timing Analysis: System execution pipeline and associated execution time of each block.

images, we considered a validation set of size 20% (with 80% used for training), corresponding to 170 images. This image set accounted for a total of 415 brake light objects, as each vehicle has 2 brake lights and each image may have multiple vehicles. The mAP is computed as the mean of the average precision (AP) values for different Intersection-over-Union (IoU). The IoU specifies the fraction of the total object ground truth area being covered by the output of the detector. Considering True Positives (TP), False Positives (FP), and False Negatives (FN), the AP value is calculated by computing the area under the curve in the precision vs recall graph, where,

$$\textbf{Precision} = \frac{TP}{TP + FP} \qquad \textbf{Recall} = \frac{TP}{TP + FN} \tag{2.1}$$

We report the AP for IoUs ranging from 0.5 (50%) to 0.95 (95%) in steps of 0.5, and the mAP as the mean of these APs, in Table 2.6. We report the precision vs. recall graphs, for IoUs 0.5, 0.75 and 0.95, in Figure 2.11. We determine that the mAP value for our brake light detection model is 90.253%. We observe from these results that the brake light detection model is effective with an average 90% accuracy, and maintains this accuracy consistently

for IoUs up to 90%. We posit that such a high mAP for such high IoU values validates the efficacy of the model and attests its usage for brake light detection in the wild. Through our analysis of the model's performance at IoU of 0.95, We observed that the sharp trade-off between precision and recall for IoU=0.95 was due to the large number of FP and FN. We observed that the high FP were due to the model confusing with other red color and circular objects in the scene, and the high FN were due to the model missing the brake light due to small size or partial occlusion or blur in the brake light pixel areas. We observe monotony in these ROC curves because of the size of our data set being in the order of a few thousands (around 3500), which we are currently scaling. The red lines are the interpolation curves. We posit that such issues can be addressed by expanding the dataset and including diversity in brake light types, for example, by imaging brake lights of vehicles in other Asian and European countries.

Camera Receiver Packet Reception: We evaluate the efficacy of camera communication receiver through the PER metric, which is the ratio of the number of erroneous packets to the total number of transmitted packets. We compute the PER over a complete experimentation trial session. We regard a packet error when none of the two packets in a transmission session of two seconds, are received with 0 bit errors, and regard a success when at least one out of two packets are received with zero bit errors. In addition to the PER, we also consider the average hamming distance (AHD) and the average decimal difference (ADD). We define AHD as the average number of bit errors in the erroneous packets, and ADD as the average of the decimal value error in the warning ID decoded from the erroneous packet's payload.

We evaluate the packet reception performance for two setups: (i) stationary, where the two LED stop lamp transmitters and camera receiver were stationed on tripods and separated by a constant distance of 10m, and (ii) mobile, which represents the real-world scenario captured in our experiments with the LED placed on a car top and a camera placed on a follower car.

Setup	**Dist.**	**# Packets**	**PER**	**AHD**	**ADD**
1. Stationary	10m	1053	2.5%	0.93	2.55
2a. Mobile	<10m	55	5.54%	0.72	2.18
2b. Mobile	<20m	103	6.5%	1.28	3.85
2c. Mobile	all	123	7.1%	1.39	4.07

Table 2.7: Camera receiver packet reception performance. For the mobile cases, the vehicle speed was within 15–40mph, and involved 10% of the overall cases for the transmitter vehicle to be positioned in the neighbouring lane.

We report the packet reception performance for these two setups in Table 2.7. We observe from this table that, the packet error rate (PER) is within 7% for distances up to 10m in static and mobile conditions. A PER of 7% is an acceptable number for practical real world usage, and comparable to radio based (e.g. DSRC (55)) packet reception performance. We also can observe that the PER is within 7% for real world mobile conditions when one vehicle is following another vehicle within a lane at safe distances. We posit that the PER can be reduced by improving signal demodulation by learning the channel conditions and adapting thresholds based on the machine learning model for the vehicular channel. The efficacy of the packet reception is further highlighted in the AHD and ADD values. We can observe from these values that system overall suffers from 1-2 bit errors on an average, and is within 4 decimal values. We hereby infer that the packet reception can be made (close to) error free by mapping the warning IDs with at least 4 decimal value differences. This means that

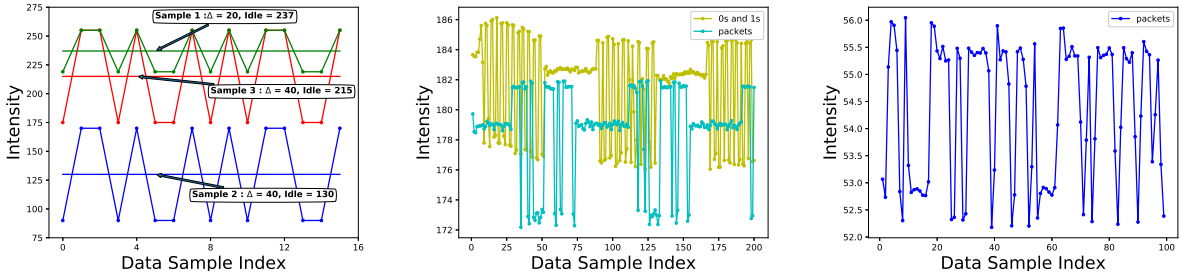


Figure 2.13: Sample signal snapshots for Δ selection microbenchmarking. Transmit signal (left), Transmit and Receive packets (middle), and Received signal on camera image pixel for 100 frames (right).

Distance	**# Packets**	**Accuracy**	**Scene Perception error**	**Communication error**
<10m	55	93.4%	2.23%	4.45%
<20m	103	90.9%	3.94%	6.5%
all	123	90%	3.02%	7.1%

Table 2.8: Real World Trace Based System Evaluation

the number of IDs that can be encoded with the current 5-bit payload will be limited to 16. This is not a problem for our current system implementation which targets nine warning IDs.

2.1.8.3 Real World Trace-Based System Evaluation

We evaluate the performance of our NLOS perception system based on the error in the final recommendation at the receiver and the total response time of the system. Our trace-based experiment involves the scene perception and transmission from one car, and reception on another (follower) car, in real time.

Accuracy: We note the received recommendation is correct or in error based on manual annotation of the appropriate safety recommendation for each visual in the recorded real-time footage. We report the error percentage in Table 2.8 for different range of distances,

and also dissect the errors from scene perception and communication. Overall, our system is able to perform with accuracy at about 90%, and up to 93% when the distances are within 10m range. To the best of our knowledge, we are not aware of any system that can successfully communicate safety signals in visual occlusion scenarios with 90% accuracy at 4 car distances (a car is about 4-5m long). We also observe that the errors are largely due to the camera communication, which can be improved, as mentioned earlier, using machine learning approaches.

Timing Analysis: We report the timing analysis evaluation for the trace-based system pipeline in Figure 2.12. This timing analysis is a result of the average value of execution times of each block in the system pipeline from 2500 trials. We can observe that the total response time starting from scene perception up to recommendation generation of the system is 1.3248 seconds. This is the maximum response time on our implemented prototype.

In the proposed work, we use 100 frames for camera communication receiver module to receive 1 second worth of LED signal. Therefore, generating a recommendation leveraging the fused line-of-sight and non-line-of-sight information takes (1 + 0.3248) seconds for the system. Please note, none of the modules (1) or (2) are bottlenecked by the LED-Camera communication delay (1 second). Clearly, the line-of-sight recommendation is sufficient to alert the driver about any eminent threat in the vicinity. Though, a recommendation about the non-line-of-sight critical event takes 1.3248 seconds, however, the delay does not limit the performance of our system. We describe the phenomena further below.

As discussed in the section 2.1.5, our system propagates a single warning that encapsu-

lates the emergency/safety critical driving situation at any given point of time. A generated warning associated to a time period, refers to the observation duration at which an agent (human/intelligent system) observes the interactions among road users/signs and gather inferences. To collect the list of inferences an agent (human/machine) needs to observe scenes for a period of time. On that account, our warning/recommendation generator system is fundamentally not limited by the LED-Camera communication delay, rather is limited by the observation-inference collection duration. Lane change behavior detection takes around 6 seconds on average, according to NHTSA (56) standards and Tomer et. al. (57), traffic light detection - nationwide(USA) travel time to stop line at start of yellow is between three to six seconds, and red interval is one to two seconds, total change interval is between four to eight seconds (58). In our system, both the lane change category's observation time falls between one and eight seconds, which gives our system some buffer time for LED data transmission-reception.

We note that it is possible to reduce the response time by performing the reception and processing in parallel. This is possible by enabling random access memory (RAM) share between the ZED camera interface and the processing application in the Jetson Xavier GPU board. There is no support available currently for the Jetson board to perform such memory share. We envision that this may become possible with future versions of mobile/robotic GPU boards.

Distance	Avg. Intensity	Dynamic Range
1 m	233.5	27
2 m	223	24
5 m	215	18
10 m	50.5	3

Table 2.9: Received signal digital intensity value from camera receiver image pixels.

2.1.8.4 Microbenchmarking: Δ selection

We briefly discuss our empirical evaluation to select the Δ value in our prototype implementation. We recall that Δ is the effective change in intensity between the transmit HIGH and LOW. We conducted experiments where the LED stop lamp was set to transmit a stream of 25 bits corresponding to one packet, in a repeated cycle, at 50Hz. We captured the footage of the LED stop lamp using the ZED camera, at 100 FPS, at different distances. We first filtered the Δ based on the flicker effect perceivable by human eyes. We observed from our experiments with 4 graduate students (age $<$ 30 years) and 1 faculty member (age $\geq$ 30 years), that the maximum value of Δ was 40. Δ values below 20 were flicker-free, however, resulted in intensity differences between HIGH and LOW at 10m within 1 in pixel value. We then used Δ of 20 and 40 in the transmissions, across different idle values for transmission. Based on packet demodulation errors, we observed that Δ =40 is the best choice for our system. Considering Δ =40, we report the received signal intensities and the dynamic range between HIGH and LOW on the receiver, in Table 2.9.

In Figure 2.13 (left), we report sample signal snapshots of the Δ modulated transmitted signal (left) intensities at different values of Δ. We can observe that a smaller Δ reduces the dynamic range between highs and lows, which eventually makes receiver demodulating more challenging. In Figure 2.13 (middle), we report a sample transmit and received packet.

The graph shows the intensity of the transmitted packets and camera pixel intensity of the same packets at the receiver. We can clearly observe the differences between the highs (1s) and lows (0s) in these signal snapshots. In Figure 2.13 (right) we report the camera pixel intensity values of the received packet over 100 frames recorded at 10m. In our system, we take the average over a 10x10 block of pixels on the camera image for processing to reduce noise in the demodulation and estimation processes. We can observe that the highs and lows are reasonably separated in the pixel domain thus facilitating robustness in the threshold based demodulation process.

2.1.9 Conclusion

In this paper, we introduced a novel architecture for NLOS perception for road vehicles, that enables a virtual see through an occluded vehicle functionality. Our system uses a dashboard stereo-camera to perceive the scene in front and communicate that through visible light communication to a follower vehicle. We have designed and implemented a proof-of-concept prototype of a NLOS perception system, that can enable identification of nine safety events, corresponding to traffic light statuses (red, yellow or green), other vehicle merge behaviors, and pedestrian presence. As a part of the scene perception design, we have trained YOLOv3 for vehicle brake lights detection using transfer learning. Through experimental evaluations on real-world driving camera footage and real-time trace-based testing, we demonstrated that our system is able to identify occluded events up to 90% accuracy and sustain communication packet error rates in less than 7%. The key use-case of our system is for driving assists, by notifying about safety and time critical events which falls under the level-2 autonomy

definition for autonomous vehicles. Such a feature can be very helpful when integrated into a fully-autonomous driving vehicle, which plants seeds for the future work emanating from this research. The models and experiment data from this paper have been set for open-sourcing.

2.2 A Cognitive Information Processing Pipeline for Multiple–Access

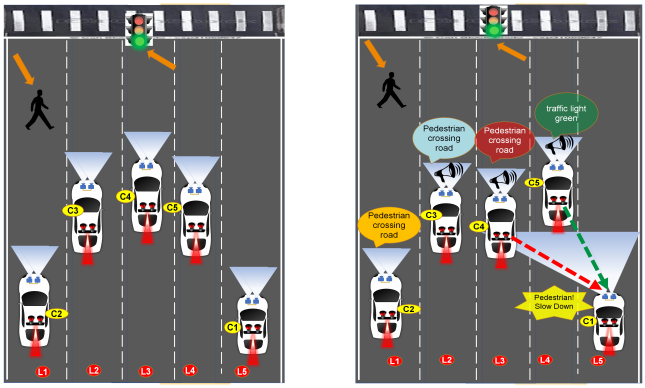
Proactive identification and prediction of safety critical events on road can be very beneficial to both, human driver assistance as well as autonomous driving. It is intuitive that most of the accidents that occur on roadways are primarily due to the fact that the events in front of the vehicle were unexpected or that they are realized very late leaving too short time for a plausible safe response (or reaction). This largely happens because the critical event is not in line of sight (view) of the driver/vehicle, as the vehicle is far from it and/or that it is being obstructed by other vehicles on the road.

Our recent work aimed at addressing this problem through our proposed concept of non-line-of-sight perception (2); of being able to see-through the traffic on roadways by integrating camera scene perception with LED-camera communication. We will refer to this prior work as see-through-vehicle for the rest of this paper. In the see-through-vehicle system, each vehicle acts as a transceiver which perceives the scene (using a camera) and relays information about any eminent emergency/urgent situations to its surrounding vehicles using LEDs. Vehicles receiving this information blend it with their local perception and generate a recommendation for a cognitive action to the driver/vehicle. The see-through-vehicle

system basically augments the local information perceived by the smart sensing units of a vehicle. The additional benefit is the accrue of information through the relaying process from neighboring vehicles.

Let us consider an example driving scenario to understand the importance of NLOS perception. Consider the road vehicular scenario in left sub figure in Fig. 2.14. Here vehicle C1 sees C4, C5 in its field-of-view (FOV), however, the pedestrian in lane L1 is occluded to C1, but C1 has the green light in view. On the contrary C4, C5 sees both the traffic light and the pedestrian crossing the road. In real life driving situations, ideally C2, C3, C4, C5 lowers the speed to let the person cross the road even if the traffic light is green. However, C1 which is unaware about the situation may not decelerate and collide with the person crossing the road. This unexpected surprise may lead to a critical accident both for the pedestrian and the driver/vehicle. Using the see-through-vehicle system concept, this situation can be avoided, if C1 could hypothetically visualize/perceive the occluded part of the road. This can happen with the help of independent/combined information received from C3, C4 or C5 that describes the safety-critical event 'hiding' in the occluded road area.

In our prior work (2) we addressed the non-line-of-sight perception over a single-access scenario where the information is communicated only from the front leading vehicle. In this paper, we extend the idea of see-through-vehicle functionality across a multiple access scenario where, potentially, all neighboring vehicles communicate information to a follower vehicle independently. We posit that all the independently conveyed information will be combined along with the vehicle's own local camera perception data. In this regard, in



(a) Local road driving scenario.

(b) Listening to neighbors

Figure 2.14: Demonstration of how a potential accident can be avoided through non-line-of-sight (NLOS) communication among neighboring vehicles.

the rest of this paper, we present the logical design of the cognitive information processing system that addresses the NLOS perception for multiple access cases using vehicular camera communication. The right sub figure in the Fig. 2.14 shows an illustration of how multiple access NLOS perception can be useful for the scenario presented in the left sub figure in Fig. 2.14.

2.2.1 Novel Contributions

In summary, we make the following key contributions in this paper:

1. We design a prediction-attention module that extracts key gists from the road scene. The module performs prediction on the receiver vehicle's local camera perceptions and then applies an attention selection approach to identify the most critical event. We refer to this selected event as the gist of processed scenes.

2. We design a decision-making module that fuses several received event information from transmitter vehicles and the extracted gist from its own prediction-attention module, to

declare the alert event. Using this alert event as the reference, the decision-making module further makes the appropriate recommendation for action to the driver/vehicle.

3. We present an analysis of information flow through a cluster of vehicles. We focus on, (a) how scene perception information of a road area is prioritized and propagated, (b) how received event information from multiple sources depicts different obstructed road sections including precise relative positions of other vehicles and event objects, and (c) how to prioritize which road section poses higher critical events.

4. We discuss the applicability for our proposed cognitive information processing pipeline by case studying three unique scenarios.

2.2.2 Related Work

Background. In our previous work (2) we designed and implemented a NLOS perception system considering a single access channel setup that enables a receiver vehicle to learn about the safety-critical situation from a single leading transmitter vehicle. A participant vehicle hosting the proposed prototype contains a controller, a LED-Camera communication module, a deep learning based scene perception module, and an intelligent information mapper. A controller, hosted in every equipped vehicle, compiles and transmits a concise packet (bits) describing a single safety-event that it generates by the information fusion of host vehicle's scene analysis and received wireless optical signal (safety-event packet) from the leading transmitter vehicle. The intelligent information mapper does the information fusion. In this work, we extend the single transmitter-receiver channel to a multiple-access channel that

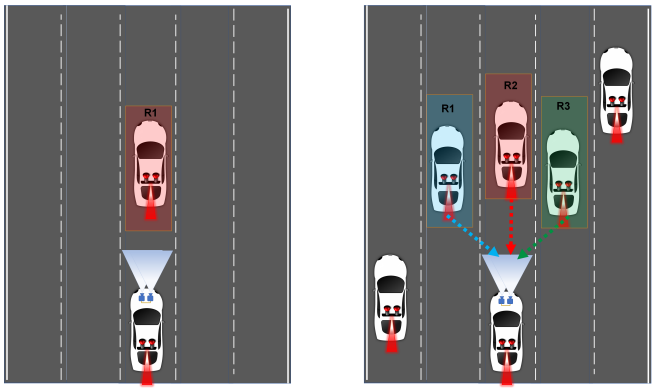
ensures simultaneous reception of LED signals in a single camera receiver. Our new system contains an enhanced road scene gist extraction module that relates to the human perception and gist recognition process, a decision-making module that fuses several received event information from transmitter vehicles and the extracted gist event from receiver vehicle's perception and finally declares the alert event. We also render a scene information flow analysis through a cluster of mobile vehicles. The information flow analysis shows interesting aspects of understanding the information propagation in the obstructed road areas.

Multiple Access Vehicular VLC. Prior work (59) performed a simulation study to reduce multi-user interference through SDMA for vehicular VLC using adaptive frontlight system (AFS) based lead headlights. This is a selective approach to choose the best performance LED among a matrix, to achieve least interfered communication. In (60) authors designed a VLC system using color-shift-keying (CSK) modulation and code-division multiple-access (CDMA) technology simultaneously. In (61) a synchronization algorithm for the CMOS camera was developed to improve the symbol rate of multiple-access two-way visible light communication. In (62) the design enables multiple LEDs with a static camera to communicate via the non-line-of-sight links. Prior work in this space as represented above, largely focus on the static transmitter-receiver setup and invest solution development for increasing bandwidth through multiple-input multiple-output (MIMO) techniques. On the contrary, our proposed system does not mandate the MIMO requirement and provides a complete solution that captures the process of information flow from multiple moving transmitter vehicles to receiver vehicles by leveraging single-access peer-to-peer (P2P) links intelligently. Our

system enables the scheme of understanding the entire road scene ahead including visible and invisible areas, by logically stitching all the relevant received information from multiple (accessed) transmitters (vehicles).

Scene Information Propagation. The work in (63) follows an epidemiological propagation model and explores message spreading nature among a large number of vehicles during certain traffic congestion (accident, rush hours). However, this does not include propagating actual scene data comprising of object positioning details in a concise format, such as in our proposed system. The work in (64) conducts scene segmentation and object detection for traffic scene understanding with deep learning, (65) uses multi-view geometry and deep learning to enable to localize and perceive the environment. Authors in (66) implement simultaneous object detection, depth estimation, semantic segmentation to understand a road scene. These representative works that use deep learning machinery focus entirely on the driving and road scene perception limited at the object-detection level only. Other works conduct the so-called gist recognition (67; 68; 69) concept for extracting the scene understanding. In our work, we have extended the concept of gist recognition by linking position and interactions between individual road objects. This enables our proposed system to understand a road situation from a collected sequence of snapshots over a period of time and predicting eminent future. Then it propagates the information to following vehicles in a compact packet structure through LED-Camera channel.

In the area of connected vehicles using 802.11p standard wireless communication, (70) proposes a safety ensuring system to help decide driver to start a safe overtaking based on



(a) Single Access (b) Multiple Access

Figure 2.15: Single access scenario (left) and multiple access scenario (right).

the traffic conditions. The drawback of the system is it requires high bandwidth for real-time video streaming of the obstructed scene to the following cars. Our proposed system, however, notifies about several emergency events to the following vehicles through a compact small data packet in the VLC channel.

2.2.3 Overview

For a vehicle entity, augmenting the information from an otherwise occluded or NLOS scene primarily demands investigation on how to acquire knowledge about the unseen area. A single (P2P) or group of witnesses (multiple-access) may tell about what they see. From the information what a host vehicle gathers by viewing and listening, it needs to prioritize and find the next forthcoming recommended action for the driver as well what to tell following vehicles. This forms the baseline requirement for the proposed cognitive information processing system. The cognition also requires (a) integrating mechanisms to listen and announce (tracking and identification, communication channel and protocol modeling), and (b) understanding the received messages from witness and preparing comprehend announcements to

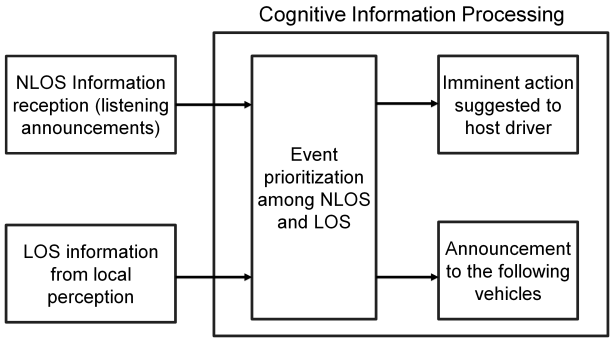


Figure 2.16: Overview of proposed Cognitive Information Processing Pipeline

the following listeners (message decoding and encoding respectively). In our previous works (2) and (1) we have primarily addressed the tracking and identification, communication channel and protocol modeling, message decoding and encoding building blocks of the system. In this work, we present the theoretical validation of cognitive information processing module for multiple-access vehicular camera communication. We keep the prototype implementation as a future work. Fig. 2.16 depicts an overview of proposed cognitive information processing pipeline.

Acquire Knowledge of the NLOS/unseen Area. A vehicle which does not see a specific section of the road receives information about the real-time events from the neighbor vehicles, to those the obstructed area is visibly accessible. There can be immediate neighbors who directly witness the occluded area and inform about that to the following neighbor(s). On the other hand, the immediate neighbor can notify about an event that it may not able to perceive directly but has been informed about from its preceding neighbors. The notion of transmitting contextual information involves both single-node and multi-node broadcasting to meet above mentioned transmission requirements. A single access scenario shown in the

left sub figure in Fig. 2.15 is composite of a single witness and a single listener vehicle. In the communication context, the witness vehicle is the transmitter, and the listener vehicle is the receiver. Conversely, a multiple access scenario shown in the right sub figure in Fig. 2.15 consists of a single receiver, that receives data from multiple transmitters simultaneously. The data distribution method is broadcasting. Since the audience is dispersed and the goal is to propagate the scene information like an announcement so that the message reaches to as many as possible receiver entity at the same time. The data transmitting method is serial, a single bit one after another, at a time.

Multiple-Access in Vehicular Camera Communication. Simultaneous reception in camera communication is possible without any collision as the transmitter (LEDs) are spatially separated in the camera view. In the scenario shown in the right sub figure in Fig. 2.15, multiple LED transmitters are transmitting to the single receiver. Each transmitter zone in the image can be divided into separate region of interest (ROI), shown in different colored shaded regions in this figure. As the channel is multiplexed spatially, multiple simultaneous communication can happen if we can identify each transmitter separately. To ensure a seamless complete transmission cycle a transmitter need to be tracked over the active duration of the transmission session. The identification and tracking can be done through a DNN. Since the communication is one-way, there is no acknowledgement/feedback from the receiver. However, this is not an issue as processing and assimilation of information at the receiver is performed asynchronously.

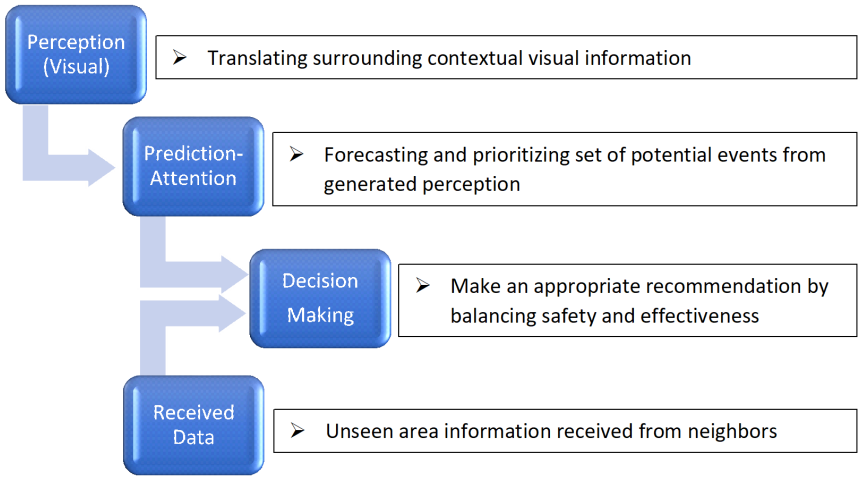


Figure 2.17: Functions in the cognitive information processing pipeline.

2.2.4 Cognitive Information Processing

Human drivers gather as much as possible sight from their vision for a very short period and then apply cognitive processes like prediction, and attention (71) on that accumulated sight. This way one can discard few event's occurrence possibilities (prediction) as well as adopt few events as urgent (decision-making), and eventually take an appropriate action based on the inference. Our work essentially translates this sequence of human cognitive processes into machine intelligence as per the key functions in the cognitive information processing pipeline depicted in Fig. 2.17: (a) perception, (b) prediction-attention, (c) received data and announcement, and (d) decision- making. We note that the perception function has been already handled in our prior work (2), and we will address the other three functions in this work.

```
{
          any_self_lane_front_vehicle      : FALSE,
         speed_limit     : 65,
         current_speed     : 50,
          traffic_light_red     : FALSE,
          traffic_light_green     : TRUE,
          merging_traffic_right     : TRUE
}
```

Figure 2.18: An example of key-value mapping generated from scene perception.

2.2.4.1 Prediction-Attention

The perception module yields a list of detected objects with their corresponding position and state values (i.e., 3 dimensions, relative speed for a mobile object like a car). That list is ideally a mapper in the key-value format. That does not make a sense of scenario or a real-time situation. The Prediction-Attention aims to generate all tentative situations from that mapping. 'Prediction' uses stored information to guide the interpretation of forthcoming sensory events, and 'attention' prioritizes these events according to their behavioral relevance (72). A sample example of perception key-value mapping is shown in Fig. 2.18.

Intuitively we see that, *any_self_lane_front_vehicle* stating the current riding lane of the host vehicle is clear. *speed_limit* and *current_speed* is denoting that the host vehicle is riding below the allowed speed limit. *traffic_light_red* and *traffic_light_green* is denoting that the traffic signal is also clear to pass through. All the attributes separately indicate different driving conditions. Intuitively, in this case, a driver would deduce that the road is clear. However, the last attribute in the key-value map *merging_traffic_right* is indicating that a vehicle coming from right is attempting to merge. Though there are multiple indications of safe acceleration however, if the last attribute value is ignored there is a high possibility of

```
announcements: [
{    neighbor_id    : 1,      pedestrian_crossing     : TRUE,     threat_dist    :
    18,    relative_lane   : -1    ,     hop_cnt    : 1}
{    neighbor_id    : 2,      pedestrian_crossing     : TRUE,     threat_dist    :
    20,    relative_lane   : -2    ,     hop_cnt    : 1}
{    neighbor_id    : 3,      traffic_light_green     : TRUE,     threat_dist    :
    15,    relative_lane   : 0",     hop_cnt    : 0}
]
```

Figure 2.19: List of announcements received from 3 different transmitting nodes.

collision. In summary, every single attribute in the mapping list is a guidance to a unique prediction of a tentative event. In the next step, the attention function prioritizes among all the predictions.

2.2.4.2 Received Data (Announcement)

Now, assume a list of announcements shown in Fig. 2.19 are received from neighbors. A single row contains neighbor_id, identity of a neighbor for a period based on the physical location relative (distance, packet arrival time) to the host vehicle. Here, pedestrian_crossing, is the critical unseen event from the host vehicle. threat_dist, is the distance between the announcer and the detected event object. For the first record in the announcement list the neighbor vehicle is 18 meters apart from the detected pedestrian jaywalking.

The relative_lane denotes the tentative position of the detected object. This attribute provides direction of road areas where the event may occur. $relative\_lane : 0$ indicates same lane, negative and positive values indicate left and right lanes respectively starting from the transmitter vehicle. Also, the lane position can be traced down by updating the value based on the transmitter's riding lane. An instance shown in Fig. 2.20; a vehicle detects an event object in the 1st lane to the left from its own riding lane has $relative\_lane : (-1)$. Similarly,

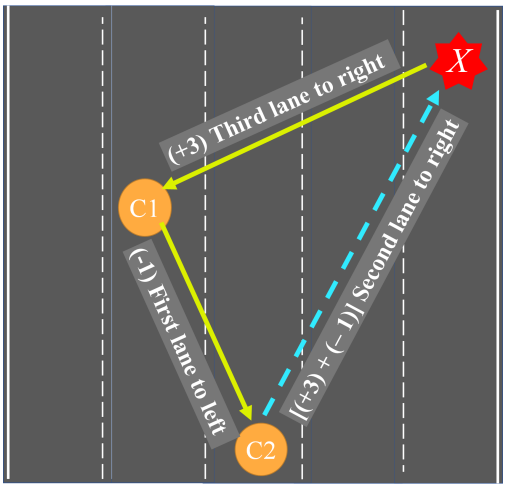


Figure 2.20: Tracing down the actual relative lane number.

an event object detected in the 3rd right lane from its own riding lane has *relative_lane* : 3. The vehicle transmits the values as it is to the following vehicles. Now, the receiver vehicle receives the announcements from the transmitter and also localizes the transmitter relative to its own riding lane. Then the receiver retrieves the actual relative lane position of the object by adding the offset lane count. As an example, the transmitter C1 localizes the object in the 3rd lane to the right from its own riding lane. C1 transmits *relative_lane* : +3. Then C2 receives *relative_lane* : +3 and localizes the transmitter that, C1 is riding 1st lane to the left of C2's own riding lane. Hence the offset lane count is (-1). Therefore, the final *relative_lane* for C2 to the object X is $[(+3)+(-1)] = +2$, that is 2nd lane to the right that is the correct position of the object X.

2.2.4.3 Decision Making

The actions to be taken by a vehicle significantly depends on the actions of other vehicles in vicinity. The situation gets more complicated when the traffic is denser, or when some

of the road area is occluded by neighboring vehicles. Prediction-Attention functioning alone cannot produce a final recommendation for action. Decision making balances among potential recommendations to secure safety and efficiency. Though the host vehicle possesses some understanding about the surroundings from its local perception, however, we posit that following vehicle's actions are impacted by the leading vehicles. Therefore, the contextual information about unseen area from preceding vehicles may have dominance over the primarily adopted local recommendation.

For example, mapping values from local perception {any_self_lane_front_vehicle: FALSE, speed_limit: 65, current_speed: 50, merging_traffic_right: FALSE, merging_traffic_left: FALSE} allows the host vehicle proceeding forward by accelerating under a valid speed limit. However, from the announcements (Fig. 2.19) the host vehicle would override its previous decision of acceleration, instead it would start decelerating and be mindful of the jaywalking pedestrian.

2.2.5 Algorithm Design

2.2.5.1 Iterative Prediction-Attention

Prediction. The perception outputs form the baseline input for the prediction module. This module predicts the future state of the vehicle using the detected object's (from camera images) current status or pose, depth from the observer camera, observation time and velocity of the observing vehicle. For instance, a future location of a detected pedestrian in a consecutive sequence of scenes can be predicted from that person's walking velocity and moving direction. Similarly, with a known velocity the observer vehicle's future location

after a certain time span can also be estimated. Now to deduce if the pedestrian and observer vehicle will collide or not, the module evaluates possible hypothetical trajectories of both the pedestrian and the vehicle and try finding a potential intersection. Similarly, for an approaching traffic light, its status can be predicting from the observer vehicle's velocity, the observation duration, and usual lighting time of the light. A driving action against road signs can be predicted as well depending on the road sign information, observer vehicle's velocity. For example, Road Work Ahead sign comes with a tentative distance measurement from the location of the board. A moving over or slow down action requirement should be initiated withing the reaching time to that work zone. Given the velocity of the observing vehicle we can calculate the reaching time thus can make prediction on the necessary action.

For all detected objects with different types like road users (i.e., vehicles, pedestrians) and road signs the prediction and attention module run iteratively. In the prediction algorithm for each detection we derive the future state (i.e., traffic lights, road signs) or pose (moving objects like vehicles and humans) of the object and then predict the occurrence possibility of the event associated to that object. We chose the occurrence possibility in the scale of (very high, high, medium, low, very low). We use stored (history) information to find out the possibilities of an event occurrence. This process is similar like how a human driver predicts based on some previous experience (stored information in the form of human memory). Like, the traffic light turning from YELLOW to RED is 3 seconds and the standard deviation is +/-1 (73). Therefore, it requires roughly between mean 2 to 4 seconds. So, when a yellow light is detected by the observer for 2 seconds then we can predict approximately

after 1 to 2 (min to max) seconds the light will turn to red. For a Road Work Ahead sign, we find the occurrence probability by comparing the reaching time to the work zone. $reaching\_time = depth \times \frac{1}{speed\_of\_car}$. If the time intersects with the future timespan (T) that means the vehicle will enter the work zone in next T seconds. We estimate the reaching time with the observer vehicle's velocity and distance mentioned in the work zone alert sign. We collected the stored information from different driving analysis studies and road surveys (56; 57; 58).

Attention. While driving in a clutter scene representation in the city roads drivers interact with frequent intersections, traffic signs, human walking and cycling, and moderate to high traffic. In this situation a human driver performs scene parsing rapidly. S/he recognizes the scene gist only from couple of consecutive static visuals by performing attentional selection on temporal and spatial changes of detected objects. The attention module function is executed as follows:

1. Sort the potential events in descending order by their estimated occurrence probability.

2. Select events by a probability threshold. For example, an event of probability of 80 and above is more likely to happen in next few seconds. So, any probability of 80 and above will be treated as next occurring events.

3. Perform gist recognition on the selected events. That is, find out 1/2 significant events those must be attended to avoid any unsafe situation.

4. Now the gist recognition is performed based on some pre-stored information. Every event is associated with an action. The action list is (Acceleration, Deceleration, Left, Right). All

the selected events will be labeled with an action in this step.

6. Deceleration has higher priority over acceleration. Now we calculate the weightage of acceleration and deceleration events.

7. If there is any deceleration event then we find the reaching time to the threat (both way when both the threat and host are approaching each other and there is a possibility of collision).

8. If any of the deceleration events has lower reaching time than any of the acceleration events, then the deceleration event is the safety critical event for that session and it must seek the highest attention.

2.2.5.2 Decision Making

The decision-making module is responsible of choosing the highest time critical event among multiple critical events. The host vehicle has some primarily prioritized events from its own perception-attention. Note that every transmitter communicates only a single highest priority event during a transmission session. However, a vehicle also receives multiple safety events from leading vehicles. Among all these events the vehicle chooses which one is the most attention seeking event. Then it recommends the host driver itself and relays the event to its following vehicles through the transmitting unit.

Mobile nodes travelling to the same direction in near proximity form a semi static network among them. To accomplish the data transmission, transmitter nodes need to be in the FOV of the receiver nodes for a certain amount of time. This notion helps to imagine a cluster

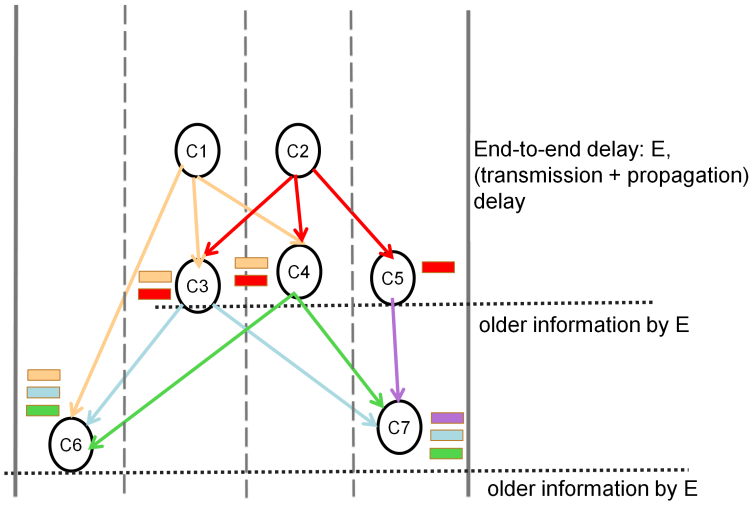


Figure 2.21: Real-time vehicular topology. End-to-end delay for a single session is E. An event information reaches to a receiver node by E time. Therefore, a received announcement is older by E time.

of interacting nodes for that span. The nodes are vehicles, and edges are data transmission links. In the Fig. 2.21 we depict a cluster of moving vehicles in a multi-lane road section. The topology follows Directed Acyclic Graph (DAG) properties. Boxes in different colors represent data packets carrying an event information.

Prioritizing among the received events help provide better safety alerts. Each received events from different directions indicates unseen situations of different sections of the road and surroundings. For instance, for C7, data coming form C5 is describing the critical situation happening straight-ahead in the road, on the other hand data coming from C1 is providing contextual information about the left-ahead corner of the road section. A cognitive combination of the events from different directions may help achieve a bird eye view to system. That eventually helps to pick up the safest and time critical event. Fig. 2.21 also depicts how information reaching down to the cluster hierarchy becomes older by the end-to-end delay from parent to child node.

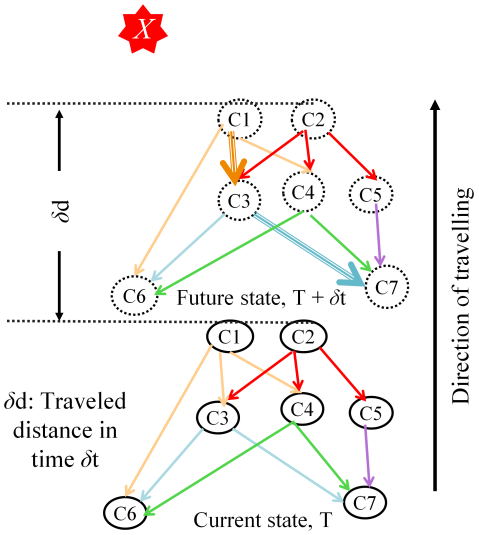


Figure 2.22: Illustration showing future state of group of cluster nodes.

The red star marked as X in the Fig. 2.22 is assumed to be a critical object (user/sign) that potentially creates a time critical event. C1, C2, and C6 localize X by their individual perception modules. However, the rest of the nodes are unaware about X. C1 and C2 create a packet and transmit to the following nodes, C3, C4, C6 and C3, C4, C5 respectively. The end-end delay for the packet to reach to the receiver node is assumed to be E. In the LED-CAMERA setup we assume end-end delay refers to the summation of ($transmission + propagation$) delay.

Any received event information is older by an order of E units of time. We represent the delay as, $n \times E$. Here, n is the edge count of the path followed by an event notification that is originated at the witness node. For instance, C3 learns about the object X by a delay of $1 \times E$. Here, the notification is originated by C1 and the path followed by the transmitted event is $C1 \rightarrow C3$. Similarly, C7 learns after delay $2 \times E$, where path is $C1 \rightarrow C3 \rightarrow C7$. Now the cluster is assumed to be moving entirely by δd distance during δt time.

As in Fig. 2.22, at time T, C1 initially calculates the reaching time to X that is t_i. At time $(T+\delta t)$ C1 moves δd distance closer towards object X. At that point, the reaching time from C1 to X is $t_j = t_i – \delta t$. Now, assume the topology remain static of the cluster and a critical event notification travels from C1 to C7 through the path $C1 \rightarrow C3 \rightarrow C7$. Therefore, C7 is now aware of object X. From C7 to X the *reaching_time = ((reaching_time C7 to C3 + reaching_time C3 to X) - Event travel time* $[n \times E]$*)*. The δt refers to the Event travel time $[n \times E]$. The packet received by C7 contains accumulated distance from $C3 \rightarrow C1 \rightarrow X$. In this case, from the velocity of the receiver vehicle C7, and the accumulated distance in the received packet we can derive the total reaching time.

The receiver node prioritizes among the announcements based on calculated reaching time, event occurrence location, and safety notions. For example, for C7 among 3 received announcements in the Fig. 2.19. The traffic_light_green is near proximity event as it has the least reaching time among the other two announcements. The relative lane for traffic lights is always 0 as the lights applies to all the lanes in the same direction of travel. And *relative_lane* : 0 denotes the event impacts the same lane where the receiver node is travelling. However, in spite of having longer reaching time the first two records in the list must seek higher attention considering the safety notion. Two relay nodes are announcing *pedestrian_crossing* that they listened from witness vehicles. Relative lane values denote, single/more pedestrian/s object is tentatively detected in the left side area of the road and the object's location falls in the danger zone (that is (+/-)4 lanes). So, the highest attention seeking announcement among the list is that, one/more pedestrian located in the right

side around reaching time of (18m + distance from receiver to the immediate transmitter – $2 \times E$)/speed of receiver.

The critical event chosen by the decision-making module is also relayed to the follower nodes as the highest priority event. A following node, however, picks the critical event based on its local perceptions a received announcements following the same process.

2.2.6 Case Study

We present the applicability of the cognitive information mapping for different use-case scenarios. We consider these case-studies as the subjective evaluation of the applicability of our positioned design. We describe the examples with respect to the host (receiver) vehicle.

(Case 1) A local road cluttered intersection consisting pedestrian, cars, traffic lights. Our three-step cognitive information processing system first performs prediction based on local perception output. We suppose, the local perception module detects traffic signal lights, neighboring vehicles. On the collection of detected objects, we run prediction algorithm that assigns the occurrence probability to each potential event, which is the likelihood of occurring an event within next T span of time. Suppose potential events are traffic light remains green/yellow (probability: very high), merging vehicle to the lane (medium), a road work after the intersection (very low), given the probability stamps are, very high, high, medium, low, very low. Now we run the attention algorithm to select the high priority event form the locally perceived likely-occurring event among all the potential events. The merging traffic and road work gets ruled out considering their higher reachability distance thus traffic light gets selected that indicates a clear road to pass through the intersection. Now, we dive

into the decision making with the received announcements from the neighboring announcer vehicles. We assume, the received safety alerts are, *a:* clear road (means, no blocking traffic), *b:* merging traffic to right adjacent lane, *c:* pedestrian in the second lane from the left. Now, the reaching time to the received event objects (traffic light, pedestrian, merging traffic) and the relative position from the host vehicle to the pedestrian is derived with the available information in the announcements. Suppose the alert a with $relative\_lane : 0$ indicates riding lane is clear to accelerate. b passively indicates a safe acceleration too as the merging is happening one lane to the right (+1), but not in the same lane. However, the alert c coveys a deceleration with $relative\_lane : (-3)$ as a pedestrian fall in the danger zone that is detected inside (+/-) 4 lanes. We assume, c gets picked up as the highest priority event. Lastly, comparing the locally selected event, clear road and the received information from neighbors, that is, pedestrian crossing road, our system chooses the pedestrian alert to act upon for the host vehicle driver as well announce to its neighbors.

(Case 2) A traffic signal light obstructed by a large truck in the local road. Perception module detects objects like neighboring vehicles, road signs. The prediction derives potential events like clear road based on perception output like no detected merging traffic, or front vehicle riding in safe distance. Now, applying the attentional selection, the locally adopted event is clear road. However, from the leading vehicles, the host gets a traffic light red event. The host vehicle deduces the traffic signal position from the received information. Thus, the host vehicle adopts the traffic light red event over clear road as the recommended action. Therefore, the vehicle gets notified about the obstructed red signal

and take proper action to avoid any threatening consequences

(Case 3) A sudden crash/brake/slowing vehicle in the adjacent lanes in the highway. In the highway, the perception module detects mostly neighboring vehicles and their interactions. Assume, from the local prediction-attention the host vehicle deduces merging traffic from the adjacent lanes on one side of the road. It also receives announcements from the leading neighbors about a sudden crash/brake/slowing vehicle in certain section of the road. From the locally selected merging traffic alert the vehicles either can decelerate or may move to an adjacent lane. However, with the help of continuous received announcements the host vehicle eventually gets an alert message to move to much safer lane from the imminent crash or blockade.

2.2.7 Conclusion

We positioned the design and use-cases of a cognitive information processing model pipeline for driving assistance that generates spontaneous safety alerts by understanding and prioritizing surrounding LOS and NLOS information. We designed a gist recognition algorithm comprising of prediction-attention concepts. In the prediction module we estimate all event occurrence possibilities based on the locally perceived scene data (LOS information). The attention module prioritizes among the probable events and select the most critical one. In addition, we designed a decision-making module that prioritized and chooses the most critical safety event based on the information from neighboring vehicles. We validated the applicability of our design through subjective case-study analysis of three real-world life safety threatening driving situations.

CHAPTER 3

CAR MOTION ANALYSIS: MEASUREMENT STUDY ABOUT VEHICULAR RELATIVE SPACIAL MOTION USING STEREO CAMERAS

The extremely high bandwidth and directionality of Optical Camera Communication(OCC) presents numerous opportunities for high throughput communication with high spatial reuse. These features make OCC an interesting case for vehicular networking using brake lights and head lights as transmitters, and optical/image sensing devices as receivers. In particular, vehicular OCC can benefit from high throughput directional links between vehicles and infrastructure (e.g. downlinks and uplinks between vehicles and road side edge/cloud computing units). The high spatial reuse factor can enable multicasting and multiple–input multiple–output (MIMO) communication (e.g. a OCC network of platooning cars, visual MIMO (9) for vehicles). However, OCC lags behind radio frequency (RF) communication in terms of adoption as a key vehicular networking technology. This is attributed to the fact that OCC links are highly directional, making them highly susceptible to throughput reduction and link failures during mobility. Therefore, realizing vehicular OCC in practice fundamentally requires addressing mobility.

OCC links require strict spatial alignment between the transmitting and receiving optical elements. Such an alignment becomes extremely challenging with traditional optical receivers that use photodiodes, due to the very small cone of reception or field–of–view (FOV). The FOV in a OCC system can be increased by using a lens in front of the receiving elements, however, this makes the system noisy due to the additional collection of ambient light noise at the receiver. The noise can be spatially filtered, while maintaining a large FOV, using

camera inspired receivers due to the image sensor's pixelated spatial structure. Selective filtering of ambient noise can help increase the receiver signal–to–noise ratio (SNR). The pixelated structure enables spatial resolvability (filtering) of ambient noise and the large FOV provides a larger angle of freedom for mobility for the transmitter–receiver pair. This way, camera inspired receivers present unique advantages to address the mobility issue. However, addressing the mobility problem requires a clear understanding of the amount and types of motion that the vehicular OCC system may encounter.

Prior works in vehicular OCC have largely been limited to theoretical concepts or controlled experiments in primarily static or controlled mobility settings. Only a few works in recent times that have explored OCC for vehicular communication in realistic mobile settings. Shen et al (74) present their pilot studies on using a photodiode receiver and a brakelight LED transmitter for low data rate communication on real highway driving settings. The study reveals the need for a better understanding of the instantaneous motion of vehicles on the road to help locate the transmit beam and retain link alignment. Yamazato et al (75), conduct a motion characterization study under V2I and V2V scenarios using a LED array and a high–resolution camera. However, the experiments were conducted in a very controlled setting and did not capture the broad range of realistic vehicular movement on roadways. Additionally, the speed of the vehicle was limited to about 18-20 miles–per–hour (mph), which limits the scope of mobility characterization knowledge in context of vehicular networking.

3.1 Motion Characterization for Vehicular Visible Light Communications

To gain a better understanding of mobility in vehicular OCC, in this work, we present a holistic study of the motion of the vehicles in real world driving settings. We take an approach where we make extensive measurements, using a OCC setup on real roadway driving, and derive our insights based on the observations from analyzing the measurement datasets. Our experiments involve a colored chesssboard pattern marker fixed on a lead vehicle and two (identical) cameras fixed on a following vehicle. We analyze the camera images and estimate the amount of movement of the vehicles in each of the three spatial dimensions X, Y and Z (see Fig. 3.9 (a)) in typical driving conditions. In essence, this paper presents an empirical study of motion in vehicular OCC by measuring the geometric effects on camera images, due to the relative motion between transmitter and receiver. We note that motion also leads to photometric effects such as signal quality reduction, pixel intensity reduction, motion blur etc., which we reserve for future work.

While prior work has explored mobility in vehicular OCC, to the best of our knowledge, our work presents the first extensive study of motion of vehicles in uncontrolled real–world driving scenarios. The dataset in our study is comprehensive of about 15 hours of video footage at 30fps, which translates to analysis of over 1.5 million images (sample points). The dataset associated with this work will be made available to the community to further the development of research in mobile vehicular OCC.

(a) Vehicle coordinate axis (b) Transmitter setup (c) Receiver setup

Figure 3.1: Vehicle coordinate axis convention and experiment setup involving the lead (transmitter) and follower (receiver) vehicle

3.1.1 Novel Contributions

In summary, the key contributions of this paper are as follows:

1. Definition of a motion model describing vehicle motion in 3D, measurement of vehicle motion in 3D, and a study of the relation between vehicular motion behavior (type of motion) and degree of motion through the model.

2. An extensive real–world experimentation involving multiple hours of data collection in realistic vehicular driving conditions.

3. A verification analysis of the inferences by applying the derived motion on a random dataset. Fairness is maintained in the evaluation by ensuring the dataset is mutually exclusive to the one used to derive the model.

3.1.2 Related Work

Vehicular VLC has been gaining increasing interest in the research community in recent times. Several works (10; 18; 76; 77; 78) have proposed techniques for improving reliability in vehicular VLC through using redundancy from LED arrays and/or using image sensors for

receiver diversity. (79) proposes the use of a imaging based control channel for tracking the LED transmitter and assisting communication on a narrow FOV photodiode receiver. Such works attest to the fact that image sensors play a key role for tracking assistance and signal quality enhancement in vehicular VLC systems. Prior works on tracking LED transmitters in vehicular VLC systems have largely focused on addressing the mobility problem for niche use–cases. Such techniques have largely used a reactive approach, where the motion has affected the quality of the signal and the research aims to address the after–effects in signal quality.

Our work aims to address motion proactively, by first gaining a comprehensive understanding of the degree and type of motion in vehicular VLC systems. We will use this fundamental understanding to further develop efficient transmission and reception protocols for vehicular VLC. To this end, (80; 75) are the only works in recent times that have approached to modeling motion in vehicular VLC. However, the works have been largely limited to specific driving settings which impedes the generalization of such studies/models.

There is significant prior literature on the use of multiple cameras for depth estimation using stereo vision in vehicular systems (81; 82; 83). The techniques range from using disparity images to sophisticated 3D euclidean point reconstruction. We note that our work does not claim to innovate on stereo vision algorithms. Our work presents a motivational use–case for multi–camera setups in vehicular VLC systems, and can piggyback on the advancements in stero vision depth estimation techniques.

3.1.3 Motion Model and Experiment Setup

We map the mobility characterization problem to the estimation of amount of motion of the vehicle in 3D space. In a wireless communication context, it is the relative motion between the transmitter and receiver that impacts the signal quality. Hence, we focus on the problem of estimating the relative motion between a transmitter vehicle and a receiver vehicle. Estimating the relative motion translates to the problem of determining the relative positioning between the two vehicles at any instance of time.

To this end, we characterize the motion of vehicles through a motion model that describes the relative positioning between two vehicles (transmitter and receiver) along three spatial dimensions. We define motion model as a vector,

$$\underline{M} = [\delta_u \;\; \delta_v \;\; \delta_w \;\; n] \tag{3.1}$$

where, δ_u captures the relative movement along horizontal (X), δ_v captures the relative movement along vertical (Y), δ_w captures the relative movement along the driving path (Z), and n is a flag value representing the vehicle behavior class type. Here, the motion model is described for a single time snapshot. The relative movement values represent the motion over a specific time window and the vehicular behavior class is the type of motion the vehicle undergoes in that specific time window. We will define and discuss the vehicle class types in detail in Section 3.1.4.3.

3.1.3.1 Challenges in vehicular positioning

Positioning requires precise location information in 3D. Using global positioning system (GPS) coordinates of each vehicle we can determine the distance between the vehicles or the relative position of the vehicles along Z dimension. However, there is no information on the other two (X and Y) dimensions. Also, GPS is prone to errors in the typical range of 3–10 meters and upto 100 meters under poor signal reception regions. Such a degree of error, can significantly detriment the quality of a VLC link as the errors can lead to losing track of the transmitter and/or communicating with the wrong vehicle; for example, a typical lane on a highway is 2m in width and a 3m error will imply a different vehicle in the next lane.

Relative positioning between two vehicles on road is also extremely challenging due to the random driving behavior of the vehicles. This implies the receiver must be able to predict and/or estimate the type of transmitter vehicle motion behavior. One approach is to fit both vehicles with inertial measurement unit (IMU) sensors that can record the amount of local motion in each of 3D axis. Since the sensors are positioned on each vehicle, the transmitter will require to inform the receiver of the sensor value. Such a necessity creates a chicken–egg type problem, as the prime reason for exploring vehicular VLC is to establish communication between the two vehicles. While using a control radio channel to share the sensor data is a possibility, this does not scale well in realistic driving conditions. Also, IMU sensors drift with time, making them not a suitable choice for precise motion measurement.

Due to the challenges in using GPS and motion sensors, we choose to study motion in a vehicular VLC link using an optical wireless setup. In particular, we indirectly measure the

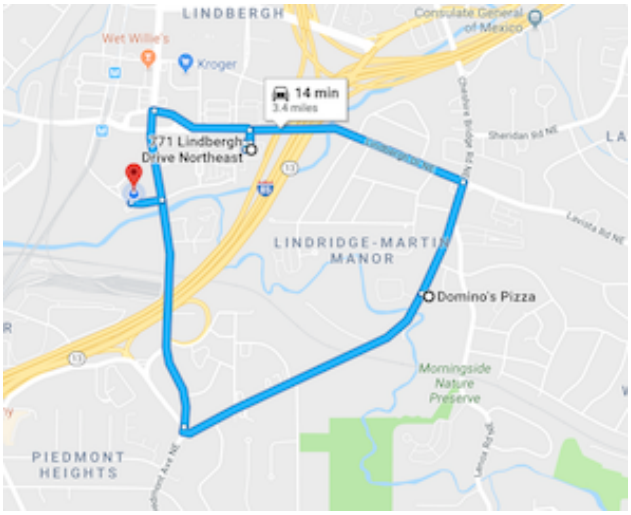


Figure 3.2: Driving roadmap of experiments conducted. The picture shows the local road pathway (speeds 25–45mph). The parking lot data (speeds 5-25mph) was collected on a 30m x 100m parking lot in the location marked by the red pin on the map.

amount of vehicle motion by studying the movements in the image sensor pixelated domain.

3.1.3.2 General Experiment Setup

The measurement study involved conducting experiments by driving two cars along different types of roads in the city of Atlanta in Georgia, USA. During the experiments it was made sure one car followed another car. Fig. 3.9 shows the experiment setup along with the devices placements and axis conventions used in our experimentation.

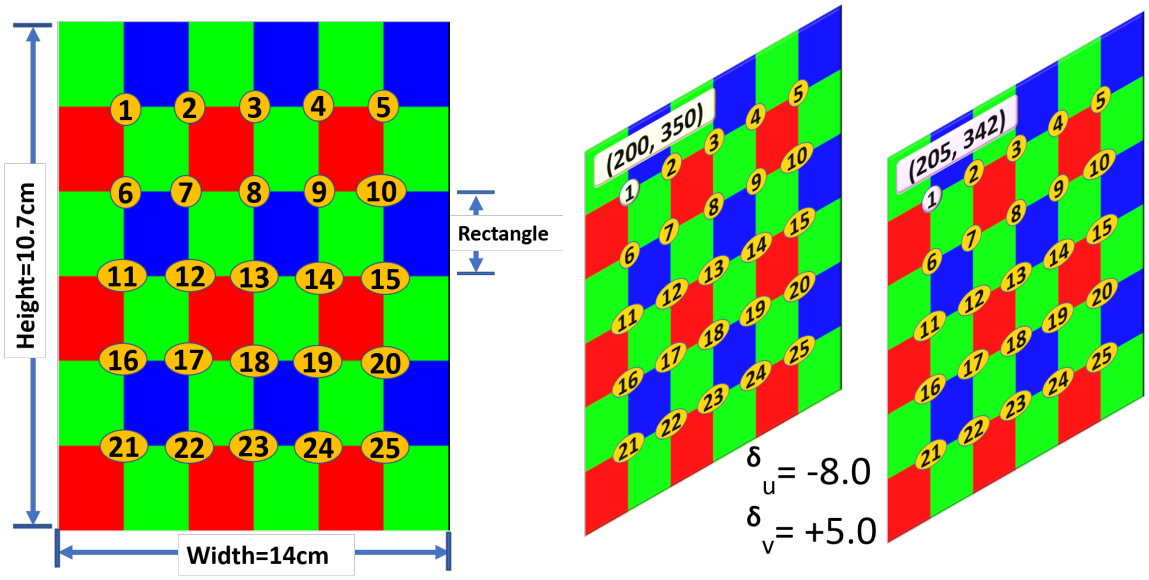
A color chessboard presenting a Bayer BGR pattern was pasted on the back of the lead car. The chessboard is treated equivalent to a static–valued light transmitter. The car was followed by the receiver car that stationed two GoPro 5 HERO cameras. The two cameras were placed at a distance of 40cm along the horizontal (X) and zero relative height difference. The image view planes of the cameras were aligned parallel to one another. The camera was set to operate at 1920 x 1080 pixel resolution and at 30 fps. Unless otherwise specified, these

are the default camera settings in our experiments.

The two vehicles were fit with an OBD–II diagnostic monitoring device and an Android smartphone. The OBD–II recorded the GPS coordinates (latitude and longitude) and car speed. We paired the device with an Android tablet through Bluetooth and used Torque Pro data logging application, available for download from Google Playstore. We stationed the smartphone on the car's dashboard and recorded the angular rotation along 3 axis using an inertial measurement unit (IMU) sensor logging application. We consider the rotation angles along X, Y, Z axis as pitch, azimuth and roll, respectively. The frequency of the measurements were set to 1 Hz (once per second) on both devices.

The experiments involved driving the vehicles under different road conditions and driving speeds (parking lot, local road and highway). The roadmap of the experiment driving paths is shown in Fig. 3.2. During the experiments the follower car drove behind the lead car, maintaining a safe driving distance. The follower car repeated the same action as the lead car. For example, if the lead car changed the lane, the follower car also changed the lane in the same direction. During this process the cameras were set to record video footage while the OBD and smartphones logged the corresponding sensor values. Overall, we collected measurements worth of about 15 hours of video and over 50000 sensor data samples. We analyze this measurement dataset to derive the motion model by using tools from computer vision, probability and statistics and error analysis.

We follow a convention that, unless specified otherwise, all relative motion parameters are defined using the receiver (follower/camera) car as a reference. Therefore, every relative



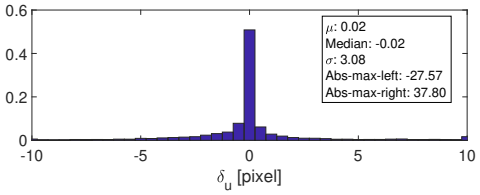
(a) Chessboard vertices (b) Sample motion computation

Figure 3.3: Illustration of motion value computation using chessboard vertices. We show an example computation of δ_u and δ_v for one of the chessboard vertices.

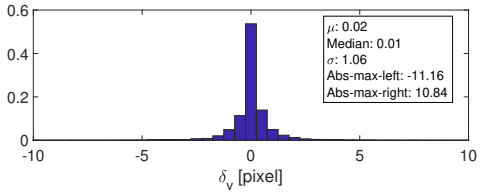
computation between the two cars will use follower car value minus lead car value. We assign positive (negative) polarity to motion when the lead car is to the left (right) of the camera center.

3.1.4 Horizontal and Vertical Motion

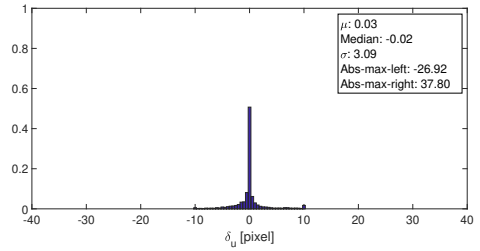
Considering the use of a camera as our measuring unit at the receiver, we denote the horizontal (X dimension: δ_u) and vertical (Y dimension: δ_v) motion parameters in pixel units. We consider, one pixel unit as the length corresponding to the side of one square pixel in the camera image sensor, set at the default resolution of 1920 x 1080. Given the camera intrinsic parameters (focal length and side length of a pixel and image sensor center), computed through camera calibration procedures (84), the equivalent amount of motion in world



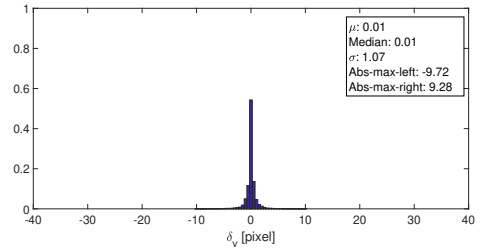
(a) δ_u 30fps



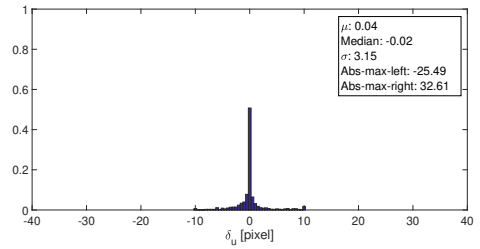
(b) δ_v 30fps



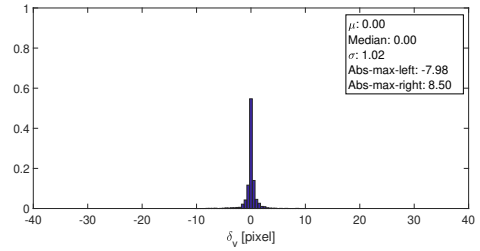
(c) δ_u 10fps



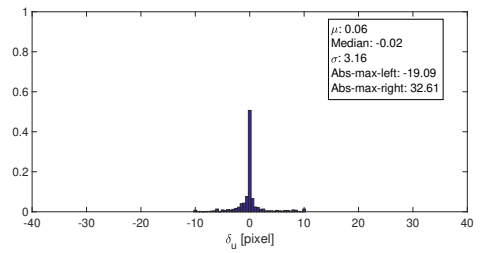
(d) δ_v 10fps



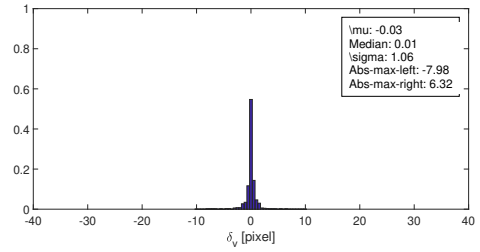
(e) δ_u 2fps



(f) δ_v 2fps



(g) δ_u 1fps



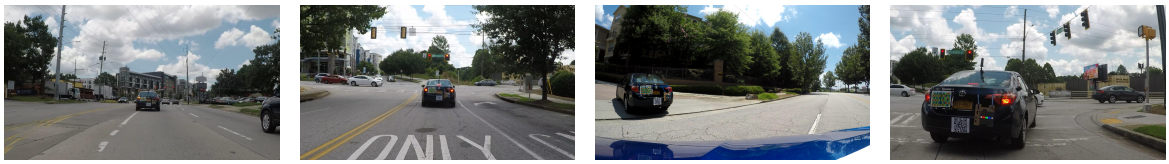
(h) δ_v 1fps

Figure 3.4: Statistical representation of δ_u and δ_v in pixels at 1920 x 1080 camera resolution

distance units (δ_u^{world}, δ_v^{world}) can be computed using perspective projection theory (85) as,

$$\Delta^{world} = \Delta^{pixels} \frac{depth}{(\frac{focal-length}{pixel-side-length})} \tag{3.2}$$

(a) SW-R (b) SR (c) SW-L (d) ST



(e) LC (f) BM (g) LT (h) RT

Figure 3.5: Image snapshots from our dataset that record each of the eight vehicle behavior types.
From top-left to bottom-right: sway right, straight, sway left, stop, lane change, bump,left turn, right turn.

Here, depth is the distance between the object and camera center. In our experiments, this translates to the distance between the transmitter and receiver, equivalently the distance between the two vehicles – denoted as δ_w. This means that computation of the relative physical movement of the vehicles in X and Y dimensions requires quantifying the movement along Z dimension (estimating δ_w). From computer vision theory, a minimum of two cameras (stereo vision) setup is required to estimate depth – using stereo correspondence algorithm (86). Depth can also be estimated from a single camera using structure from motion algorithms (86), however, that requires knowing the exact type of movement of the camera, which is technically an unknown parameter in our vehicular setup experiments. The need for depth estimation motivates the use of the two camera setup in our experimentation for vehicle motion measurements. We will discuss depth estimation experiments in Section 3.1.5.

3.1.4.1 Measurements

We measure the horizontal and vertical motion using the pixel coordinates of the corner vertices of the imaged chessboard. Fig. 3.3 provides an illustration of this process. We run an open–source chessboard detection routine (87) on each image frame and record the pixel locations, (r, c), of 25 vertices (corners). We collect 25 points to provide diversity and scale the number of samples to improve the statistical estimation accuracy. If $(r_i(k), c_i(k))$ and $(r_i(k+\tau), c_i(k+\tau))$ correspond to the pixel coordinates of a vertex i (i =1,2,3..25) at time instance k and $k+\tau$, we compute the motion values as,

$$\delta_u(i,\tau) = c_i(k+\tau) - c_i(k) \quad \delta_v(i,\tau) = r_i(k+\tau) - r_i(k) \tag{3.3}$$

Here τ is the time period between each data (image) sample. Since we process every frame of the video, $\tau = \frac{1}{fps}$, where fps is the frame rate of the video.

3.1.4.2 Observations and Insights

We compute δ_u and δ_v for the entire dataset using the procedure described above, setting $fps = 30$. We also create sub–datasets by downsampling the parent dataset at lower frame rates of 1, 2 and 10fps. For 10fps we take the difference in pixel coordinates for every 3rd value in the parent dataset, and correspondingly every 15th for 2fps, and 30th for 1fps.

In Fig. 3.4 we plot the overall probability distribution of δ_u and δ_v at frame rates of 30,10,2 and 1 fps. We also provide the statistical mean (μ), median, standard deviation (σ) and the maximums in each polarity.

From the histograms in Fig. 3.4 we can observe that δ_u is bounded within an absolute value of 40 pixels and δ_v within 12 pixels. Also, we can observe that the effect of frame rate on motion values does not follow any pattern. These observations lead us to the following key insights:

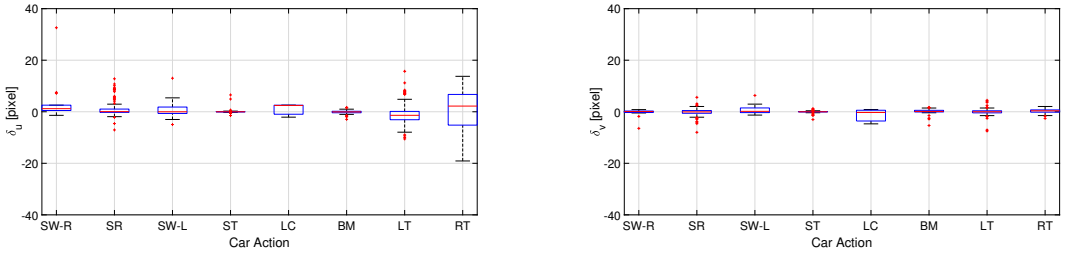
(1) δ_u and δ_v are bounded: Considering the large size of our dataset, and that our measurements are across typical road driving conditions and natural driving patterns, we infer that the absolute values of δ_u and δ_v are bounded within 40 and 12 pixels, respectively. We note that these values are relative to the camera resolution of 1920 x 1080. However, translating this number to a different resolution is simple as the it is a direct linear mapping (twice resolution implies 2x the value of δ_u and δ_v) We also note that the amount of vertical movement is significantly less than the horizontal. This agrees with our general intuition that the amount of lateral movement of a vehicle would be significantly more than the vertical. However, we do also note that the vertical movement is non–zero as it must account for the vibrations of the vehicle and also jerk movements of the vehicle due to road conditions (e.g. pothole).

(2) Vehicle movement behavior impacts δ_u and δ_v: Analyzing the motion by sampling the movements at different time windows reveals that the type of motion happening in that window matters significantly. Based on the definition of the motion parameters, the value we measure corresponds to the aggregate of the motion within the sampled time window. From

our measurements, we observe that the maximum movement is within 40 pixels, whether it is sampled at 30fps or 1fps. It would be wrong to generalize a theory that a longer time window sampling is a mere addition of the δ values in each sub–window. The cumulative effect over a window would be the case only when the movement of the car is continual across the accruing time windows. However, since that is not the only case in typical driving scenarios, we infer that that within any sampling time window the vehicle could be continually moving across a dimension or go back and forth (vehicle sways to left and adjusts back by swaying to the right in next window) or have a combination of multiple movements across dimensions (vehicle sways to left and turns right or stops). Depending on the length of the sampling window, the fine (intricate) movements may or may not be captured, and that only the position of the vehicle at the start and end of the window will only be registered. For example, if the vehicle moved to left in X by 10 pixels in 500ms and moved to right in X by 10pixels in next 500ms, and if the sampling rate is 1fps, the δ_u would be 0. However, this does not necessarily mean that the vehicle was relatively static. In this case, a frame rate of atleast 2fps can register the two events which will reflect as $\delta_u(k) = -10$ and $\delta_u(k + \tau) = +10$, respectively.

3.1.4.3 Vehicle Movement Behavior Analysis

We define 8 vehicle movement behavior actions and bin our dataset based on the movement type through manual inspection of the videos. Each action is defined as the relative movement of the lead car with respect to the follower car. We recall that the follower car follows the same action as the lead car. As mentioned earlier, the left and right conventions are in reference to the viewing direction of the camera on the follower car. The actions (see



(a) δ_u, frame-rate = 30fps

(b) δ_v, frame-rate = 30fps

Figure 3.6: Horizontal and vertical motion values for different vehicle behaviors: Swaying to right inside the lane (SW-R), Straight (SR), Swaying to left inside the lane (SW-L), Stop(ST), Lane change (LC), Bumping/brake to stop (BM), Left turn (LT), Right turn (RT).

Fig. 3.5) are defined as follows:

1. SW-R : vehicle sways to the right within the same lane.
2. SR : vehicle driving straight within the same lane.
3. SW-L : vehicle sways to the left within the same lane.
4. ST : vehicle is stopped within the same lane.
5. LC : vehicle is changing a lane (left or right).
6. BM : vehicle experiencing a bump and/or braking to stop.
7. LT : vehicle actively turning left.
8. RT : vehicle actively turning right.

In Fig. 3.6 we draw the boxplots for the measured δ_u and δ_v at each of the 8 vehicle behavior types.

On δ_u we can observe that the variance of the motion values within a particular class/type is different across the 8 classes. We observe that behaviors involving turn type movements (lane changes, active turning) have a higher variance than driving along the lane. We verify our conventions of left and right by observing that RT has a median positive value and LT has a negative median value.

On δ_v, we observe that the medians and variances are fairly uniform across the types. In general, vertical movements are more of a function of the driving path topology (driving on a hill versus flat land) than vehicle behavior.

Measuring the vertical movements across different road elevations requires further experimental investigation. However, these measurements provide a significant baseline knowledge of the range of motion along these dimensions. In addition, the variability in δ_u for different behavior types motivates deeper analysis of the temporal variance of the motion within specific time windows. We believe in further analysis of the dataset can help define relevant temporal features which can be used for executing a machine learning classifier to identify and/or predict vehicle motion behaviors. We reserve such an analysis for automatic vehicle behavior learning through motion for future work.

3.1.5 Distance between the Vehicles

We measure the motion along the Z dimension, δ_w, as a function of space instead of time. In essence we measure δ_w as the spatial separation of the transmitter and receiver cars at

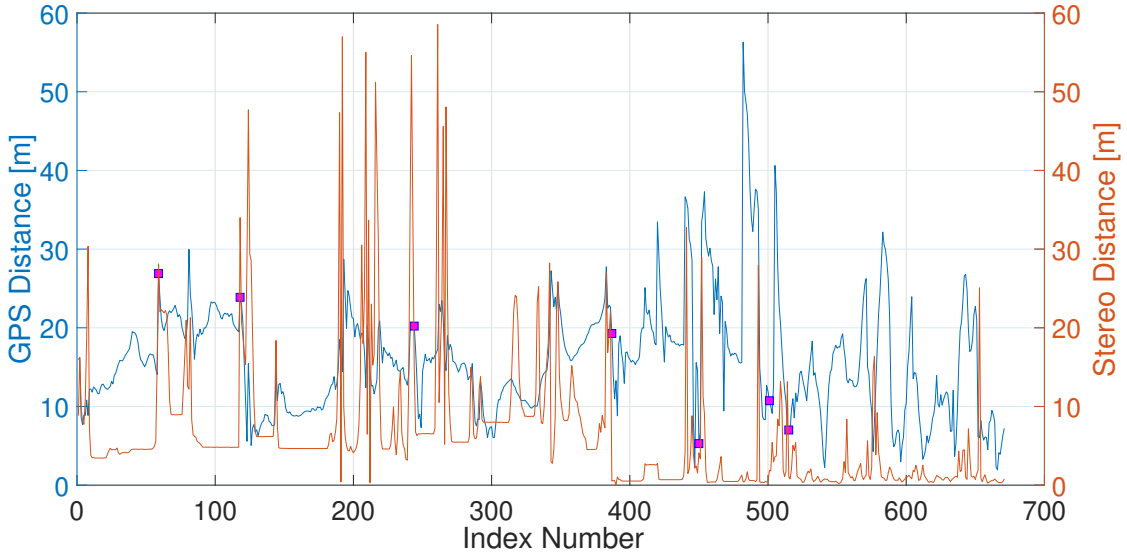
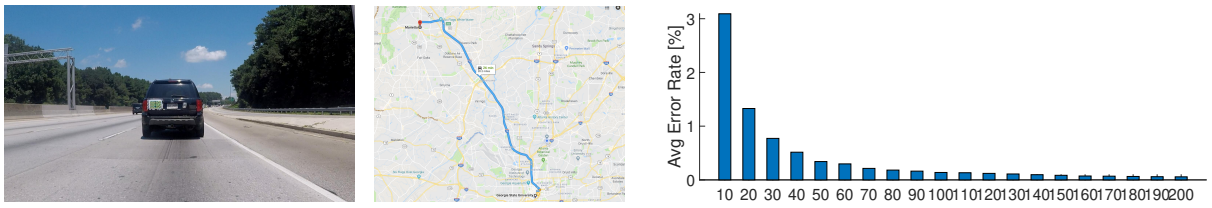


Figure 3.7: Comparison between distance (δ_w) estimated using GPS and stereo vision.



(a) Image frame (b) Roadmap (c) Error rate

Figure 3.8: (a) Image snapshot from verification dataset. The vehicles are driven on a highway (speeds 50–65mph). (b) Roadmap of the data collection. (c) Average error rate for different horizontal and vertical movement ranges.

a given time snapshot, which is equivalent to the estimating the distance between the two vehicles at each instance of time.

As described earlier, the value of δ_w is necessary to translate horizontal and vertical motion from pixels to world distance values. The translation to world distance values is important as it refers to the actual physical motion, while the pixel units refer to the motion interpreted by a camera. This means that a single value of δ_u or δ_v pixel movement will yield

a different world motion value depending on distance. However, this mapping is deterministic and linear; for the same value of pixel movement, the world movement at short distance is smaller than at long range.

We conduct a baseline measure of distance between the two cars using the corresponding GPS coordinates by applying the haversine formula (88) using GPS latitudes and longitudes.

As an alternate method for distance measurement, we use stereo vision theory for camera depth estimation (86). Given the stereo camera setup, we analyzed image frames at matching (using timestamps) timeslots from the two GoPro cameras. We used stereo correspondence mapping to match the chessboard vertices on the stereo image pair. We used 5 points in the image pairs; the points corresponded to the top left, top right, bottom left, bottom right and centroid of a virtual box bounding the rectangular chessboard.

We compute distance using stereo vision for each of these 5 points and record the mode (maximally occurring) of the 5 distance estimates. Distance is computed as,

$$\delta_w = \frac{f * b}{x_l - x_r} \tag{3.4}$$

where, f is the camera focal length, b is the baseline (distance between the two cameras in one plane), and $x_l - x_r$ is the disparity. In our setup, both cameras were aligned in X axis and thus the baseline and disparity is along the horizontal. Disparity is determined using the column (c) coordinates of the matched corresponding point pairs.

In Fig. 3.7 we report the distance estimated using GPS and stereo vision for a snapshot from our dataset. The goal of this analysis is not to propose a novel depth estimation algo-

rithm, instead is to study how good or bad are the available methods for distance estimations in our setup. We make the following observations based on the comparative evaluation of the two methods:

(1) GPS method overestimates distance values compared to stereo method.

(2) Upon manual inspection of the frames with high distance estimates (peaks) we observe that GPS method is highly susceptible to errors when the vehicle drives close to a bridge (would apply same for a tunnel as well).

(3) Stereo vision based distance estimates are closer to reality (than GPS).

(4) Disparity based stereo vision distance estimates deviate significantly when the two vehicles are at oblique angles from one another. This is because the corresponding points may not lie in parallel image planes and thus disparity is no longer applicable. The peaks marked by a red blob, in Fig. 3.7, correspond to these deviations.

(5) Stereo method underestimates distance values in some cases when the cars were very close (shortest distance between vehicles was 5m in our dataset; measured using reference landmarks on google maps).

In general, our experiments reveal that a simple stereo estimation method can help estimate distance between two communicating vehicles. However, it is to be kept in mind that stereo algorithms can have specific limitations depending on different driving scenarios and vehicle behaviors.

3.1.6 Verification Analysis

We have gained a better understanding of the degree of motion through our analysis of our measured dataset. As an action of verifying our inferences from the analysis, we apply the learning on a new dataset. To ensure fairness in the evaluation, we capture a completely new set of data, which is mutually exclusive from the dataset for motion characterization experiments discussed in this paper. We collected a total of 21 videos, amounting to about 2 hours of video footage, using a GoPro mounted on a follower car, of a lead car fit with the bayer color pattern chessboard. In this experiment, the vehicles followed each other and drove across local roads and a highway. The video frame rate was set at 30fps and resolution at 1920 x 1080. Fig. 3.8 (a) and (b) show a single image frame from our verification dataset and the drive map of the experiment, respectively.

The motive of motion characterization is to help build efficient trackers to help maintain strong optical alignment of VLC links. In this regard, we test our findings by applying a strawman tracker that fits a bounding box on the chessboard area on the image frame. The dimensions of the bounding box are set to the values of δ_u and δ_v measured from the motion experiments. We measure the accuracy of this bounding–box tracker using the ground–truth values of the coordinates of chessboard vertices. For the evaluation, we assume that the pixel coordinates of the region of interest is known on the first image of every video in the dataset.

The tracker marks each frame as a success if the heuristic bounding box retained all the vertices of the chessboard. The tracker marked it as a failure even if one of the chessboard vertex (corner) was outside the bounding box. When a failure is detected, the tracker is

reinitialized to its ground–truth position on the subsequent frame. We compute the error rate as the ratio of the number of reinitializations to the total number of processed image frames.

In Fig. 3.8 (c) we show the error rate, averaged across the dataset, for each value of the side length of the square bounding box. We can observe that the error rate is consistently significantly low ($< 1\%$) for bounding box size of 40 and above. This is in agreement with the measured range of horizontal and vertical motion from the other dataset. The bounding box tracker presents a proof–of–concept of the idea of improving tracking accuracy using auxiliary information on the degree of motion.

To this end, our work lays the foundation step towards holistic understanding of the degree of motion in realistic vehicular VLC settings.

3.1.7 Conclusion and Future Work

In this paper we presented an experimental study of vehicular motion by studying the relative pixel motion on camera receivers, which were mapped to physical world distance units. Through our experiments we generated a dataset worth over 15 hours of video footages. Upon extensive analysis of our dataset we inferred that the typical range of horizontal and vertical motion of the vehicles is bounded. We found that the bound for a high definition image resolution (1920 x 1080) is of the order of 40 pixels; which translates to about 25cm of lateral and vertical movement for a typical digital video camera at 10m distance between transmitter and receiver vehicle. We also defined 8 vehicular movement behavior classes and studied the motion values for each class. Through a strawman tracking application

experiment on a new dataset that we created, we verified that our measurements concurred with our understanding from our motion characterization experiments.

3.2 Camera and Camera Array Receiver Systems

Cameras integrated in modern vehicles are very useful resources for communication. They enable a relatively wide field–of–view (FOV), allowing more flexibility during motion through computer vision based tracking techniques. Tracking can include locating the optical transmitter or even another vehicle (treated as a different node in the network). The large number of highly directional receiver elements allow a longer communication range, as interference and noise can be selectively filtered over the spatially diverse pixel elements of the camera's image sensor. Since the pixel values are digitized and quantized, analysis and processing can be all done in software.

However, camera communication requires line–of–sight between the transmitter and receiver. This translates to the transmitter being imaged within the field–of–view (FOV) of the camera. Typical FOVs for off-the-shelf cameras are in the range of ± 60 degrees – approximately being able to image 1 lane at a 12m distance between the transmitter and camera. Cameras are also limited in their sampling rates, as typical sampling rates are of the order of 10Hz to 1000Hz for off–the–shelf cameras. The cost significantly increases for frame-rates higher than 1000fps and other custom cameras.

Multi-camera or camera-array systems can be very useful for vehicular networking applications from a multiple- input multiple-output (MIMO) system perspective (1)(89). The

additional camera can help provide meta data, for example, for tracking purposes, while the primary camera can be used for active data reception. Additionally, the multi-camera setup (virtually) expands the receiving pixel array, thus enabling a potential throughput scaling capability. In this paper, we position the idea of using camera communication for vehicle to vehicle communication. In particular, we explore vehicular multiple camera communication where light emitters such as brake/tail lights can transmit information to be perceived and decoded by cameras on following vehicles. Our experimental platform features a stereo camera setup to enable dual usage of the cameras for perception and decoding, as well as, distance estimation between vehicles. To address the key challenge for maintaining communication reliability during vehicular mobility, we estimate the movement of the vehicle through image analysis prior to decoding.

In the rest of this section, we will discuss our experimental findings and insights on vehicle movement measurements and modeling in its 3 dimensions – X (lateral or horizontal), Y (longitudinal or vertical) and Z (camera depth or distance between vehicles parallel to road surface).

3.2.1 Pilot Experiment Study using Stereo Camera Setup

As a pilot study, two cameras were deployed on a car following a lead vehicle on local roads (speeds between 30-45 mph) and highways (speeds between 50-65 mph) in Atlanta GA. Fig. 3.9 shows the experiment setup along with the devices placements used in our experimentation. A color chessboard presenting a Bayer RGB pattern was pasted on the back of the lead car. The chessboard is treated equivalent to a static-valued light transmitter.

Figure 3.9: Experiment setup involving the lead (transmitter from (1)) and follower (receiver) vehicle. The driving roadways are highway (speeds 50-65 mph), local road (speeds 25–45mph), and parking lot (speeds 5-25mph).

This transmitter setup is maintained identical to the setup used in the vehicle experiments in our prior work in (1). The key update from our prior work, that we highlight and discuss in this paper is the receiver stereo camera setup with two RaspberryPis and the corresponding analysis of the depth (distance between vehicles) related information.

The transmitter car was followed by a receiver car that stationed two RaspberryPi3 cameras. The two cameras were placed at a distance of 12cm along the horizontal (X) and zero relative height difference. The image view planes of the cameras were aligned parallel to one another. The camera was set to operate at 1920 x 1080 pixel resolution and at 30 fps. Unless otherwise specified, these are the default camera settings in our experiments.

This new stereo camera receiver setup features two RaspberryPi cameras. The cameras, integrated with the RaspberryPi module, were controlled remotely through secure shell (SSH) and to operate (capture image and video) synchronously. Time synchronization is ensured by enabling Network Time Protocol (NTP) on the RaspberryPi controllers, and parallelSSH protocol was used to operate/control the two RaspberryPi at the same time. The two vehicles

were fit with an On-board device (OBD) II vehicular diagnostic monitoring device and an Android smartphone. The OBD–II recorded the GPS coordinates (latitude and longitude) and car speed. The OBD-II device is paired with an Android tablet through Bluetooth and uses Torque-Pro data logging application. The smartphone is stationed on the car's dashboard and recorded the angular rotation along 3 axis using an inertial measurement unit (IMU) sensor logging application. The rotation angles along X, Y, Z axis are considered as pitch, azimuth and roll, respectively. The frequency of all the sensor measurements were set to 1 Hz on both Android devices. A SLAMTEC Rplidar A3 is used for ground truth distance measurements, which is a single channel, 25 meters range radius, 16000 samples per second enabled LIDAR.

The field measurements involved driving the vehicles under different road conditions and driving speeds (parking lot, local road and highway). During the experiments the follower car repeated the same action as the lead car, while maintaining a safe driving distance. For example, if the lead car changed the lane, the follower car also changed the lane in the same direction. During this process the cameras were set to record video footage while the LIDAR recorded the distance to obstacles within 360 degree range of the z (depth) axis, and other sensors logged the corresponding sensor values. Overall, we collected measurements worth of about 12000 image frames and over 10000 sensor data samples. The motion model are derived from this measurement dataset by using tools from computer vision, probability and statistical error analysis.

3.2.2 Experiment Results and Insights

Consider Δ^{pixels} as the amount of motion of a specific object in the image the pixel domain between successive frames. In our experiments this amounts to the movement of the vehicle across frames. Given the camera intrinsic parameters (focal length and side length of a pixel and image sensor center) (84), the equivalent amount of motion in world distance units can be computed using perspective projection theory (85) as,

$$\Delta^{world} = \Delta^{pixels} \frac{depth}{(\frac{focal-length}{pixel-side-length})} \tag{3.5}$$

Equation 3.5 implies that computation of the relative physical movement of the vehicles in lateral (X) and longitudinal (Y) dimensions requires quantifying the movement along Z dimension (or depth). From computer vision theory, a minimum of two cameras (stereo vision) setup is required to estimate depth using the stereo correspondence algorithm (90).

In essence, the motion along the Z dimension, or depth, is measured as the spatial separation of the transmitter and receiver cars at a given time snapshot, which is equivalent to the estimating the distance between the two vehicles at each time instance. We conduct a baseline distance measurement between the two cars using the corresponding GPS coordinates by applying the haversine distance formula using GPS latitudes and longitudes. The baseline measure of the ground-truth distance between the two cars is obtained by synchronized LIDAR distance values.

Stereo vision theory is an alternate for distance measurement in camera depth estimation. Given the stereo camera setup, image frames were analyzed at matching (using timestamps)

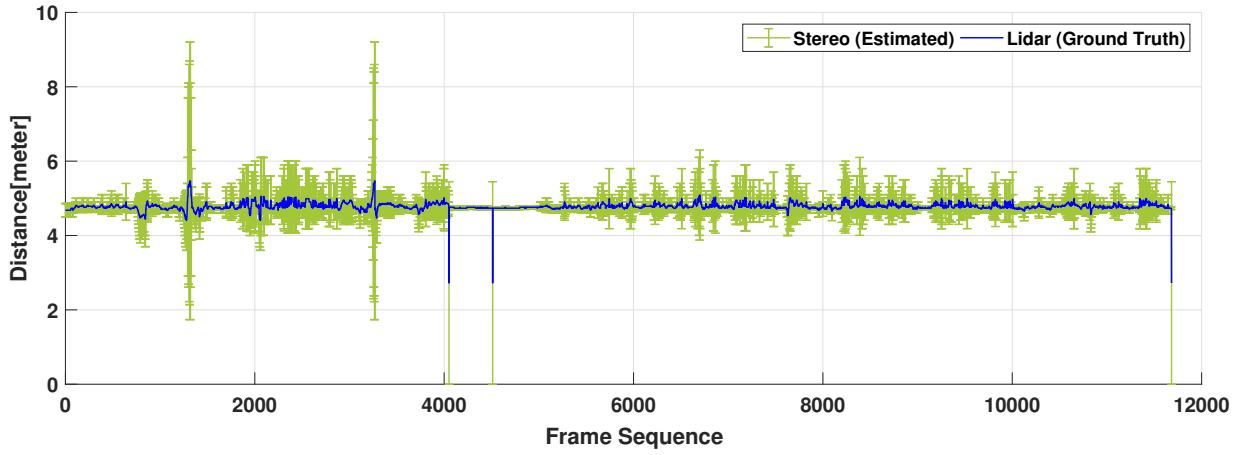


Figure 3.10: A snapshot from the dataset to show the comparison between estimated distance between vehicles using stereo vision and the ground truth values (measured using the LIDAR).

time-slots from the two RaspberryPi cameras. Stereo correspondence mapping is employed to match the chessboard vertices on the stereo image pair. Five points were used in the image pairs; the points corresponded to the top left, top right, bottom left, bottom right and centroid of a virtual box bounding the rectangular chessboard. The distance is computed by stereo vision for each of these 5 points and record the mode (maximally occurring) of the 5 distance estimates

$$depth = \frac{f * b}{x_l - x_r}, \tag{3.6}$$

where f is the camera focal length, b is the baseline (distance between the two cameras in one plane), and $x_l - x_r$ is the disparity. In our setup, both cameras were aligned in X axis and thus the baseline and disparity is along the horizontal. Disparity is determined using the column (c) coordinates of the matched corresponding point pairs.

Fig. 3.10 shows the distance estimated using LIDAR and stereo vision for a snapshot

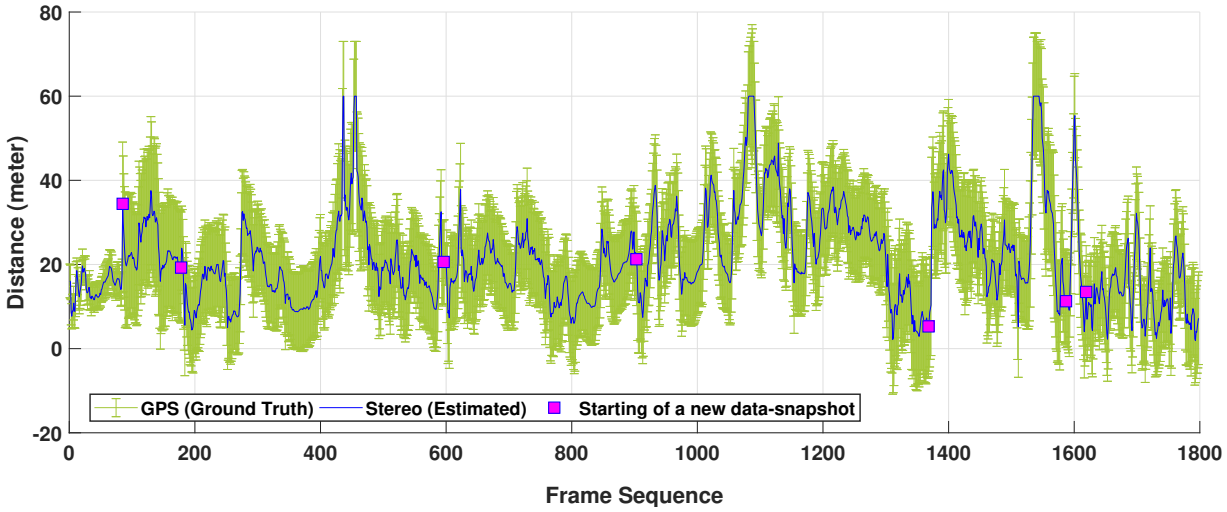


Figure 3.11: A snapshot from the dataset to illustrate the disparity between distance between vehicles estimated using GPS values and stereo-estimation.

from our dataset. The error bar graph demonstrates errors in meters in estimated stereo distance compared to the ground distance. Based on our comparison between the LIDAR and Stereo-camera estimates, we observe an error or difference between the two estimates in the order of (absolute values) 1.2m on average, 1.5m median and 6m maximum. The goal of this analysis is not to propose a novel depth estimation algorithm, instead is to study how good or bad are the available methods for distance estimations in our setup. We make the following observations based on our study:

- Upon manual inspection of the frames with high distance estimates (peaks) we observe that GPS method is highly susceptible to errors when the vehicle drives close to a bridge (same applies for a tunnel). Fig. 3.11 shows a snapshot (not shown in Fig. 3.10), to illustrate the high GPS inaccuracies in general.

- Disparity based stereo vision distance estimates deviate significantly when the two

vehicles are at oblique angles from one another. This is because the corresponding points may not lie in parallel image planes and thus disparity is no longer applicable.

- Stereo method underestimates distance values in some cases when the cars were very close (shortest distance between vehicles was 5m in our dataset; measured using reference landmarks on google maps).

CHAPTER 4

DRONE COORDINATED LOCALIZATION: PEER TO PEER AND DRONE SWARM LOCALIZATION USING CAMERA COMMUNICATION, WIFI FTM, AND SIMULATION

Autonomous unmanned aerial vehicles (UAV) or drones, are increasingly becoming popular due to their potential in achieving sensing capabilities over wide spatial areas (91). At the outset, fundamentally, all drones rely on robust GPS location for precise automated navigation during their mission flights. It is well known that GPS values are prone to errors and failures (depending on the environment). More recently it has also been demonstrated that drone GPS units are prone to a variety of security attacks (92), all the more, revealing the fragility and vulnerability of GPS on drones.

We propose to achieve the localization between the UAVs to address the problem of UAV GPS failure or its unavailability. Our proposed approach allows one UAV (helper drone) with a functional global positioning system (GPS) unit to coordinate the localization of another UAV (distressed drone) with a compromised or missing GPS system. The helper drone when discovers a distressed drone flying in proximity, estimates the distress drone's absolute GPS location information respective to its own location. This is performed by computing the relative position of the distress drone with the helper drone. The estimated location is then communicated to the distress drone. In this paper, we evaluate two fundamental approaches for range estimation for peer-to-peer relative positioning between drones: (a) camera and computer vision projection theory, and (b) WiFi Fine Time Measurements (FTM). We evaluate our proposed peer-to-peer localization via error across three estimated measures: (i)

range or distance between the drones, (ii) GPS bearing (angle), and (iii) GPS location (co-ordinates). Based on our analysis of our dataset consisting of outdoor static and flying drones setup, our methods result in a median localization error within 1-4m. We discuss this research further in the section 4.1.

We also model and simulate drone swarm localization with several distressed drones to evaluate our camera and WiFi FTM localization methods. The drone swarm modelling allows us to conserve the inherent nature of random motion and interactions of the real-world in-field experimentation while measuring the average localization error, and average execution time in the presence of an increased number of distressed drones. We conducted up to 160 simulations for 8 different parameter combinations. The simulation results attest our in-field experimental localization error ranges, and gives us a holistic idea of the end-end operation conduct timing for different number of drone participants. We discuss this research further in the section 4.2.

4.1 Peer-to-Peer Localization using camera and WiFi FTM

Failure or attack of drone GPS units can significantly compromise the mission safety – particularly, when the mission involves high value payloads. For that reason, redundancy of GPS values is important. Today, the best available method to provide redundancy is the inclusion of additional GPS modules and parallelizing GPS systems (93). Considering the limited payload capacities of drones and the corresponding trade off with size and battery lifetime, such redundancy approaches are bulky and largely impractical – especially, for

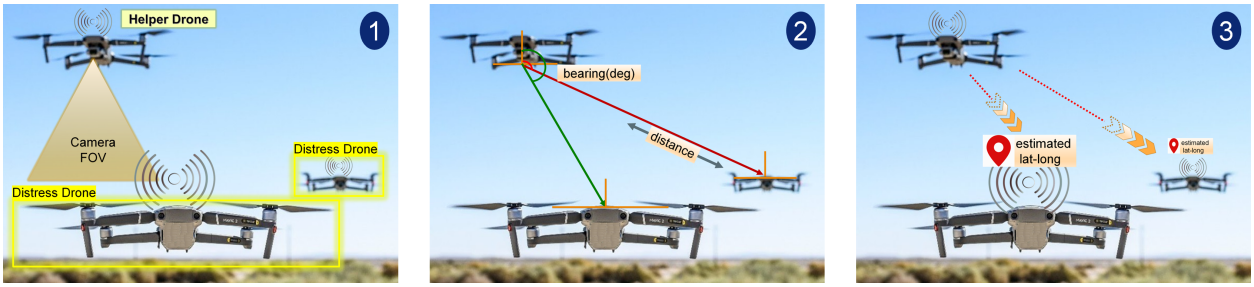


Figure 4.1: A conceptual diagram of proposed P2P DroneLoc system for remotely operating drones using on-board camera and WiFi FTM ranging devices.

medium and small drones that may not be able to mount multiple GPS units. This is because most GPS modules being used on drones have integrated magnetometers and thus have to be placed at a reasonable distance away from electromagnetic field interference from integrated electronics and metallic beams to ensure accurate and proper functionality. This way, adding GPS redundancy through the inclusion of additional GPS modules creates a space issue on most consumer drones because each module has to be placed far from other modules for magnetometers to function properly. It is due to these restrictions that most consumer drones today, capable of automated flight, only contain a single GPS module and lack redundancy.

P2P-DroneLoc. To address the GPS unreliability problem in drones, we propose an alternative solution to GPS redundancy for multi drone systems. As shown in the conceptual illustration in Fig. 4.1, we propose a design where drones with no GPS or non-functional GPS units are able to receive help to localize themselves coordinated by a helper drone with a functional GPS/location sensing unit. The key concept of our methods is a *helper drone* (HD) tracks nearby GPS *distressed drones* (DD), and then estimates the relative position of the distress drone – by estimating the distance and bearing angle. The absolute GPS coor-

dinates of the distress drone can be determined using the helper drone's GPS coordinates and the relative positioning with the distress drone by leveraging the haversine formulation.

Proposed Approach. Most consumer and commercial drones capable of automated flight have onboard cameras, inertial measurement units (IMU) for motion sensing, computing units, and wireless networking capability. To this end, we leverage cameras and WiFi APs that are already deployed commonly in drones. With advances to next generation of WiFi (WiFi 6 and beyond), there are an increasing number of WiFi devices and APs incorporating Fine Time Measurements (FTM) capabilities – uses an ACK based prototol to measure round trip time of radio signal to estimate range between two WiFi FTM devices/APs. We propose to use the FTM range estimates between helper and distress drone. For bearing or angle, we use triangulation, using a mediator drone referred to as *pseudo-HD*. The camera-based method is designed with a single on-board camera fitted on the helper drone. The helper uses the camera images to visually track the drone under distress, then applying the camera projection theory on the image the helper computes distance and bearing of distress drone. Then, using spherical trigonometry, the helper drone estimates the distress drone's absolute GPS latitude and longitude coordinates. The helper uses WiFi communication to inform the estimated location to the distress drone. In our preliminary baseline work, we have already implemented and prototyped an ACK based P2P WiFi communication to conduct inter-drone data transfer. Note that in this paper, we design, implement and evaluate the full fledged localization where our prior work serves as a preliminary feasibility study.

4.1.1 Novel Contributions

The key contributions presented in this paper are,

1. Design and implementation of single camera based P2P drone localization.

2. Implementation of a prototype system on two drones that integrates camera and WiFi localization approaches and deployment for real-world trials.

3. Extensive empirical evaluation of localization accuracy in real-world outdoor flying and static drone settings.

The rest of the sections in the paper are organized as follows: Section 4.1.2 discusses related work, Section 4.1.4 presents an overview of all the proposed localization methods, Section 4.1.5 discusses the detailed implementation of camera and WiFi localization techniques. Section 4.1.6 discusses the evaluations, and Section 4.1.7 concludes the paper.

4.1.2 Related Work

Computer Vision based target localization. There is a good body of prior work in vision-based ground target localization with a single UAV (94; 95; 96), and multiple UAVs (97; 98). Our work is helping a flying drone to achieve its location information with another single UAV. We propose an effective approach of finding a near accurate GPS information of the target UAV with help of only a single camera and with no power hungry, complex computer vision algorithm execution. A work (95) localizes a fixed target with a single camera. Authors idea of finding target location was to triangulate between the camera, the

target, and a static hypothetical reference point in the target plane itself. Camera-to-target, and Camera-to-reference point angles were measured by back-projecting LOS unit vectors derived from two pixelated points retrieved from the detected target and the reference points in the image frame. The advancement in our work is, we avoid the complexity of finding Euler angles(95) from UAV's IMU sensors rather we derive distance (2), and bearing from camera geometry and then use the Haversine Great circle distance formula (99) to find relative GPS coordinates.

UAV-assisted localization—Optical and Radio Fusion. Almost all UAV assisted localization are currently focused on search/rescue operations onto humans or onto static targets. However, when an UAV itself is in need of localization assistance, the norm is that a ground station sends help, contrary to the approach of an active UAV helping the lost UAV recovering location information. Our approach is truly distributed and uses off-the-shelf hardware and technology already available on drones. Among the few drone-assisted target localization research, (100), uses an UAV to fly over a target object and assigns that target the UAV's location value. It uses RSSI and AoA technique with several antennas to estimate target's approximate positioning. Thus (100) may generate serious erroneous positioning thus leads to an unreliable localization assistance. (101) another work proposed an UAV-assisted target localization using LoRa network for ground level IoT device localization where the UAVs act as mobile gateways. This work requires a LoRa network setup with a gateway and only localizes ground IoT devices registered to that specific LoRa network. Whereas, we develop more accurate UAV-assisted mobile target localization without any need of specific

centralized or ground infrastructure setup.

4.1.3 Baseline System Design

Our implemented baseline system considers that the drones at the basic level are equipped with a built–in flight controller (e.g. Pixhawk), GPS, barometer (altimeter) and IMU. We propose to further equip the drones with a WiFi transceiver, computing unit – such as a RaspberryPi or NVIDIA Jetson Nano board – a red LED array and a stereo–camera (e.g. Stereolabs ZED or INTEL RealSense). Each drone carries an 8x8 LED array on each of the four directions. In this design for demonstration purposes we assume two drones, one is the helper drone and another is the distress drone. The distress drones are considered not to have GPS units or that their GPS functions have been compromised or disabled. The coordinated localization approach works as follows:

- Distress drone calls for help to localize itself and a SOS radio signal is broadcasted using WiFi.
- Helper drone detects and receives the SOS signal and acknowledges the distress drone that it can help. It sends a pairing ID.
- Distress drone communicates the association ID via WiFi which once received by the helper drone initiates the localization help routine. In the meantime, the distress drone displays a 2D barcode on its LED array.
- In addition to the 2D barcode, the distress drone modulates the ON LEDs of the array using an ON-OFF Keying (OOK) pattern, where bit 1 is represented as LED-ON and

bit 0 as LED-OFF. The modulation rate is set between 30-50Hz so that it is barely visible to the naked eye yet can be captured and decoded from the frames of a video camera (following Nyquist sampling criterion) using camera communication (102).

- In the localization routine, the helper drone first identifies the distress drone visually by recognizing the 2D barcode and also decoding the bit pattern encoded in the LED ON-OFF modulation. The bit pattern is set to the pairing ID, and thus successful pairing between the drones is achieved when the helper drone decodes the ID correctly.

- Next in the localization routine, the helper drone computes the relative 3D position of the distress drone. It uses the stereo-camera's depth estimation to compute the spatial separation between the drones. It uses the X and Y coordinates of the computer vision tracking of the LED array's barcode, to determine the lateral and longitudinal (relative) orientation of the distress drone.

- Finally in the localization routine, the helper drone computes the GPS coordinates of the distress drone by fusing the relative 3D position with its own GPS, barometer and IMU sensor readings. It uses IMU sensor readings specifically to improve the precision of the localization by tracking the relative movements of the drones while coordination. This way, the coordination can be achieved with the drones in motion.

- The estimated GPS coordinates are then communicated using WiFi from the helper drone to the distress drone.

- The distress drone updates its own location by internally computing its locations using its IMU sensors and the last known location (from helper drone).

4.1.3.1 Communication Protocol

The drones establish their coordination through communication over radio (WiFi) and optical (LED-Camera) wireless channels. As described before, the helper drone pairs with the distress drone through a pairing ID code. This code is first communicated from the helper drone to the distress drone through WiFi. Then, it is communicated back to the helper drone by modulating the LEDs on the distress drone. The helper drone, once SOS signal is received, does a 360 degree rotation over the next 10 seconds to locate the distress drone by tracking the 2D barcode displayed on the distress drone's LED arrays using computer vision. The rotation is to ensure the helper drone tracks the distress drone even if it were not in its fully-frontal camera field-of-view. The helper drone uses camera communication concept (102) to decode the pairing ID encoded in the LED array's signals. Upon successful decoding of this ID, the helper drone validates the distress drone's request.

The helper drones enters into the localization routine phase after validation phase. In this phase, the helper drone first finds the relative positioning of the distress drone in its camera view using camera view projections. It further estimates the distress drone's GPS coordinates and communicates them through WiFi. This data is reorganized into a standardized format understood by the flight controller firmware (e.g. iNAV) used to pilot the drones. The distress drone sends this data to its flight controller over a UART channel using Multiwii Serial Protocol (MSP). This process is repeated at a frequency of 5Hz in order to

provide iNAV with sufficient information for mission control. Once the drone successfully reestablishes 3D hold, it responds back to the helper drone to maintain a flipped orientation and proceed with the mission while guiding the distress drone. During this phase, the distress drone also continually reports its GPS status to the helper drone. If the helper drone identifies that the distress drone has reestablished GPS connection or location then the assist protocol is stopped. However, if the distress drone fails to retain GPS/location functionality, the helper drone continues the assist protocol until it is manually terminated or until the distress drone is guided to a safe location – last mission WAY POINT or HOME.

4.1.4 UAV assisted Coordinated Localization

To ensure clarity, we revisit the key abbreviations prominently used in the rest of the paper: (a) HD - Helper Drone, (b) DD - Distress Drone, (c) Pseudo HD - Pseudo or Mediating Helper Drone.The localization assistance process is initiated with the distress drone discovery, by either a HD or by the DD. In the rest of this section, we discuss our camera-based and WiFi FTM based localization methods.

4.1.4.1 Camera-based localization with a single HD

In this method the HD reaches to a distress drone (independently or from a group) to provide localization assistance. In this setup the HD has an on-board camera and an active GPS module. The DD does not carry any camera. The location assistance process starts with detecting the flying DD in the captured image frame. An image frame indicates a point of time and a specific location in the entire duration of DD's flight. Moreover, an image frame

can provide position information like, the distance between HD to DD, and the bearing angle between HD to DD for that point of time. We synchronize image capturing timestamp with the built-in GPS (serving as HD's ground truth location). The DD's ground truth location is collected using its onboard GPS-module. The time-synchronization ensures that, we are able to estimate DD's unknown location (GPS) by estimating distance, and bearing angle from a camera image frame and also be able to validate the estimated location with ground truth location.

4.1.4.2 WiFiFTM ranging based localization with a HD and a Pseudo-HD

In this design we only use WiFi FTM ranging technology to provide location assistance to the DD. The HD and DD are each equipped with a WiFi FTM transceiver. However, in this radio based localization assistance method we need at least three drones in the scene. The three drones help setup triangulation to estimate the relative position of the DD compared to other two drone's position in the 3D world. This method is camera independent and instead introduces a secondary helper drone (pseudo HD). A pseudo-HD is in reality another DD which is location enabled through some means. We posit that, among a group of DDs a recently localized DD can be acting as a pseudo-HD and assist the primary HD in helping other DDs. The rationale for the three drone (triangulation) approach is, firstly, this method enables us estimating bearing using angle of arrival (AoA) without help of any on-board camera in any of the drones in the scene, and secondly, with the help of HD, a DD, once received localization assistance can then be denominated as a pseudo HD and can start to help other DDs in vicinity.

4.1.4.3 Location Estimation of DD

After the detection of a distressed drone we estimate the location of that distressed drone. The idea is, we can find the latitude-longitude of point B relative to point A when we know the latitude-longitude of point A, the azimuth/bearing of point B as seen from point A, and the euclidean distance between point A to point B. In the context of our system, point A refers to helper drone and point B to a distressed drone. For estimating bearing and distance we explore the potential of both the optical (camera) and radio (WiFi) channels. Note that, we get the altitude information from drone altimeter. However, changes of altitude of the flying DD does not effect our proposed localization methods. In the next section 4.1.5 we describe the mechanics of both our optical and radio based location estimation methods.

4.1.5 Localization Method

4.1.5.1 Location Estimation in Optical Channel

We design methods to estimate location of a DD using only a single on-board camera in the HD. We use computer vision camera projection theory to estimate the distance and bearing between HD to DD, then apply spherical trigonometry formulas (103) to estimate the absolute lat-long of DD.

Step 1A. Distance Measurement. [Finding distance between HD's on-board camera and the detected DD in the image frame] From the system calibration the DD's real world dimensions are known. Therefore, having the DD as a reference object on the camera image frame enables us to measure the world distance between the DD and the HD. The key

concept of estimating real-world distance between a single camera to an object captured in the image frame is, ratio of the size of the captured object on the image sensor and the size of the object in real life is the same as the ratio between the focal length and distance to the object (perspective projection theory).

Step 2A. Bearing Measurement. [Finding bearing angle between HD's on-board camera and the detected DD in the image frame] The bearing angle that is the real world angle between HD's north to the DD in clockwise direction, can be estimated from the camera image. Finding angle between HD on-board camera to the DD in the image frame leads to estimating the actual bearing angle calculation with an offset angle denoting north of the HD itself. In our hardware setup, we assume the north of the HD is pointing towards the sky, south is pointing to the ground, east to the left and west to the right of drone's forward direction, respectively. We attached our camera facing forward to cover the frontal view of the HD, considering the target object to be captured in the HD's camera is another flying drone (possibly a DD). Therefore, according to our camera orientation, we claim the angle between the camera to detected DD refers to the actual bearing angle.

The bearing angle estimation includes, (i) detecting flying DD in a captured image, and (ii) measuring angle between the image frame bottom-middle pixel and the midpoint pixel of DD's bounding box. We borrowed the drone detection module from our baseline work in the section 4.1.3. However, we implemented the bearing angle measurement in (ii) using the concept of angle measurement between camera to an object in image frame. We posit that, the image midline connecting image bottom-mid pixel (x, y) and top-mid pixel (x1, y1) is

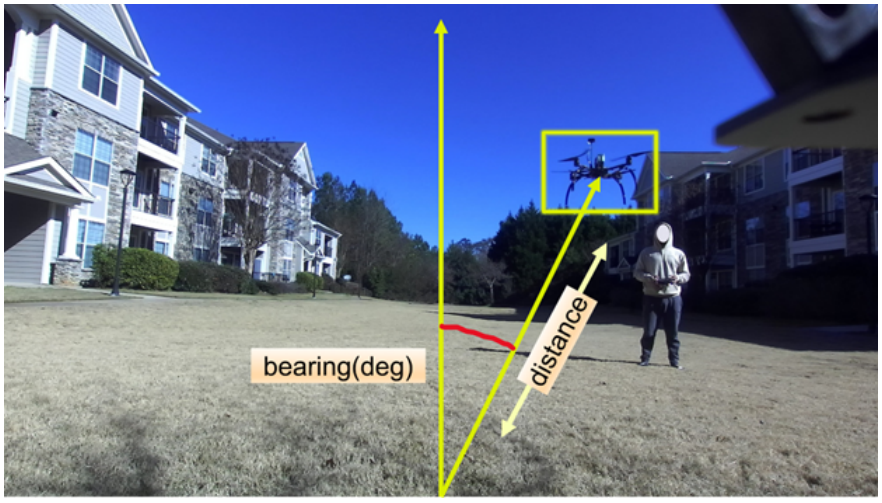


Figure 4.2: Bearing between HD to DD estimated from image.

vector $\vec{v1}$, and the line connecting the drone bounding box mid pixel (x2, y2) and image bottom-mid pixel (x, y) is vector $\vec{v2}$. We can say that both $\vec{v1}$ and $\vec{v2}$ projects onto the image center as two rays from some real world 3D points. Now, with the inverse camera intrinsic matrix and given pixels representing these vectors, those same rays can be back projected into the 3D world. Therefore, given our image top-mid pixel (x1, y1) and bounding box mid-pixel (x2, y2), we back project vectors $\vec{v1}$ and $\vec{v2}$ into 3D space. The angle between them is then computed from the dot product. Fig. 4.2 shows an image frame taken from our dataset with estimated bearing marker.

4.1.5.2 Location Estimation in Radio Channel

As mentioned before, the WiFiFTM method requires an HD with on-board AP, a pseudo-HD with on-board FTM transceiver, and the DD with another on-board FTM transceiver. We assume, the three drones form a triangle while flying. The localization process follows below steps.

- The distance between HD to DD, pseudo-HD to DD, and HD to pseudo-HD is measured using WiFi FTM ranging protocol. The distances form a hypothetical triangle where each distance between any two drones refers to a side of the triangle and the drones are vertices.
- From the law of cosine the SSS triangle is solved and the angles are measured.
- Given two known GPS coordinates of HD, and pseudo-HD, and distance between HD to pseudo-HD we estimate the bearing angle between HD and pseudo-HD.
- The HD calculates the bearing from its location to the DD's location in reference to the triangle-angles and the bearing between HD and pseudo-HD.
- Now that the HD has its own latitude-longitude pair, the distance between itself and the DD, and the bearing of DD as seen from HD itself, therefore, the HD estimates the target latitude-longitude pair of the DD using spherical trigonometry and sends along the pair to the DD.

Step 1B. Distance Measurement: [Finding distance between HD to DD, pseudo-HD to DD, and HD to pseudo-HD] We use WiFi FTM ranging protocol for distance measurements. We synchronize the on-board ground truth GPS values with these distance measurements using timestamps to validate both estimated distances and estimated GPS coordinates of DD.

Step 2B. Bearing, and Latitude-Longitude Measurement: [Finding bearing angle of DD as seen from HD, and estimating latitude longitude] After solving the SSS triangle we

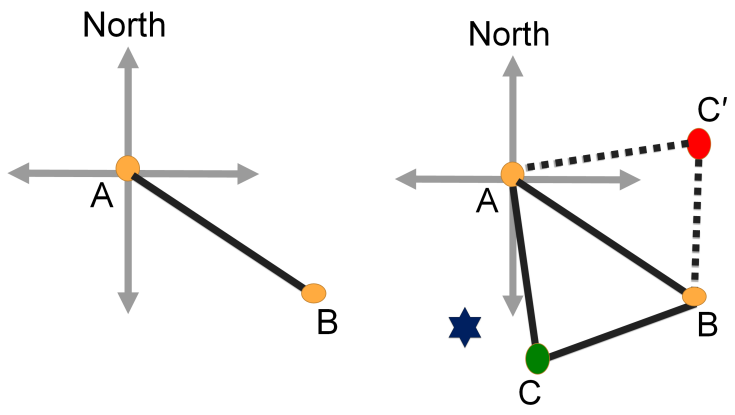


Figure 4.3: Bearing estimation of DD w.r.t HD and pseudo-HD, through fusion of triangulation and spherical trigonometry. The left subfigure presents HD, pseudo-HD w.r.t origin, the right subfigure shows possible locations of DD

have all three angles $\angle A$, $\angle B$, $\angle C$ and sides a, b, c. In the 3D space, we imagine a coordinate system, where A is located in the origin. The bearing of B can be derived from the pair of latitude-longitude of A, B and the distance between A, B that is, c. Thus, B's location in the space with respect to A can be discovered with the help of bearing and distance as shown in the left sub fig. 4.3. Now, interestingly, we have the angle $\angle A$ that is $\angle BAC$, so two possible positions of DD can be derived C and C'. One, where C is located on one side of the hypothetical line AB, or in the other side as shown in the right sub fig. 4.3. Since, we do not have any knowledge of C's bearing from either A or B, an approach to rule out a location among C and C' is to have information about DD's previous location. So, we propose and implement the aspect, by broadcasting DD's last active location coordinates. The concept is, the newly estimated location is highly prone to be closer to the previous active location. As shown in the right sub fig. 4.3 the accepted newly estimated location of DD is C assuming the blue star denotes the previous active location of the DD, thus C' is

Figure 4.4: From left HD's forward view with on-board camera, next HD's backward view while performing camera calibration, then the DD with on-board pixel 2 (WiFi FTM RTT client device), lastly a snapshot of HD filming a flying DD.

ruled out since its comparatively far from the previous location.

4.1.6 Evaluation

4.1.6.1 Experiment Setup

The hardware configuration involving proposed camera and WiFi FTM methods includes mainly (i) a single camera, (ii) two WiFi FTM client devices, (iii) one FTM supported WiFi access point (AP), (iv) Helper Drone, (v) Distress Drone, and a (vi) NVIDIA Jetson Nano computing/processing unit. For the camera unit we use footage of a single camera among two cameras of the ZED that we mounted in our HD (see Figure 4.4). For the FTM client devices we use two Pixel2 phones. Both the phones have the WiFiRttScan app installed. WiFiRTTScan app enables real-time wireless ranging between the FTM client and the AP with 1-2 meter range accuracy. We use Google Nest WiFi router as the AP. This Google WiFi supports IEEE 802.11mc FTM RTT since 2018 and it uses a single Qualcomm IPQ4019 chipset for both 2.4 GHz (no FTM RTT) and 5 GHz (FTM RTT enabled). With the two client devices and a single AP we can only get distances between any two devices, however, for some of our experiments we need distances between every pair of drones in a three-drone

setup. Hence, we use another android app WiFiNanScan that collects distance between the other two drones. WiFiNanScan app uses IEEE 802.11mc FTM RTT too. Moreover, The app measures the distance between two smartphones using the Wi-Fi Aware protocol (also called Neighborhood Aware Networking(NAN). The helper drone in our system is a basic drone with built-in flight controller, GPS, barometer, and IMU. We further install a WiFi transceiver unit and a NVIDIA Jetson Nano board to serve as a processing unit. Distress drone however in our system design, is a low-payload drone comparative to the helper drone without the GPS and any camera modules installed.

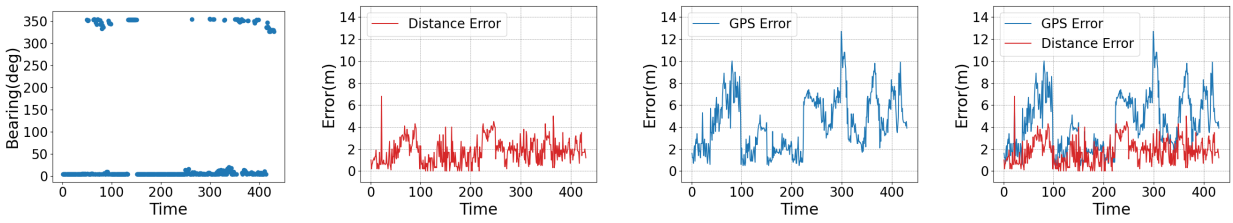


Figure 4.5: Camera-based localization evaluation results. From left, (i) estimated bearing, (ii) estimated distance accuracy, (iii) estimated location accuracy, (iv) comparison of estimated distance and location accuracy.

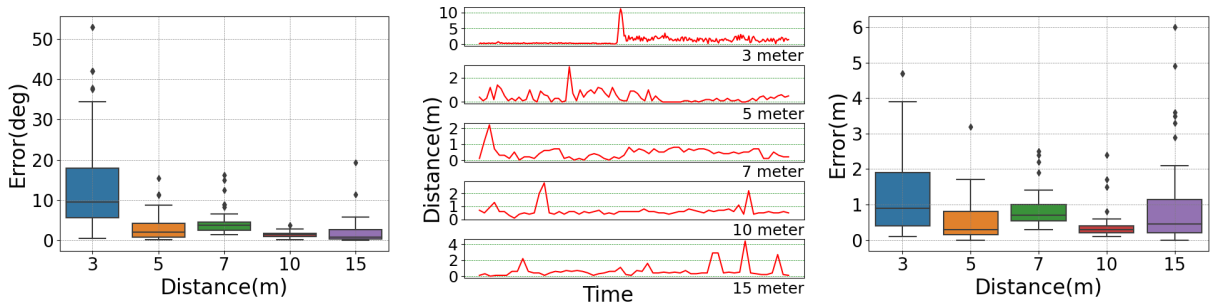


Figure 4.6: WifiFTM ranging based localization with a helper drone and a pseudo helper drone. From left, (i) estimated bearing accuracy, (ii) estimated distance accuracy, (iii) estimated location accuracy

4.1.6.2 Results

We present the experiment based evaluation of the two methods independently. We evaluate our system based on the localization accuracy based on the ranging error, angle estimation error and the overall estimated GPS coordinates error. We also present the current prototype status that integrates both camera and radio localization in an actual fully functional drone system. **(1) Single Camera based Localization Accuracy**

Setup and Methodology: Our experiments involved a single HD located static and a single DD flying in the range of 3 to 10 meters of the HD. The HD's on-board camera takes video/picture of the flying DD. The images are processed to estimate the distance and bearing angle between the HD and the DD. To validate estimated distance, bearing, and location (lat-long) we require (1) ground truth distance (GT-distance), (2) ground truth bearing (GT-bearing), and (3) ground truth GPS (GT-GPS) measurements between the HD and the DD. We measure GT-distance from GPS coordinates recorded by the HD and DD. The GPS data has dual usage, one is GT-distance, and another is GT-GPS during our evaluation. In a data collection session, a flying DD's GPS coordinates change depending on its changing positions, however, the coordinates remain unchanged for the static HD. Both the HD and DD records GPS coordinates continuously. We also marked our outdoor experiment site's ground with meter markers. These markers are an extra level of, more accurate than GPS distancing, verification for camera-to-DD distance estimation that serves us as a redundant set of GT-distances. Contrarily, GT-bearing between camera to an object can be derived by presuming a triangle formed by three points, with three sides connecting

two points on the detected object and the other point is the camera itself. Since collecting these triangle side measurements in real world units is only realistic in static setups and not possible when drones are mobile, so, for validation of estimated bearing we rely on the accuracy of the estimated location as a whole.

Metrics: We evaluate the camera-based localization for, (i) accuracy of the estimated distance between the HD (camera) and the DD (ii) accuracy of estimated GPS location of DD. The distance and location estimation accuracy is defined by error between the estimated distance to GT-GPS distancing, and estimated location to GT-GPS coordinates respectively.

Results Discussion: In Fig. 4.5 left-most (first from left), we report estimated bearing from all our image datasets. We observe that mostly the bearing values are in the range of 5° to 10° and some are between 325° to 350°. We explain these bearing values by our camera FOV and the concept of bearing measurement. According to the placement of the camera to cover up the frontal view it shots objects between 0° to 90° that is the right half of the image frame, and between 270° to 360°, that is the left half of the image frame. In the second-from-left plot we show estimated distance accuracy for all datasets, with a median of 1.5m with spread of (+/-)1m. The third plot shows estimated GPS location accuracy, with a median of 4.5m with spread of (+/-)2m. Due to drone mobility, a limitation of using GPS as ground truth is that we have to be prepared to deal with noisy ground truth data. This explains the reason for the median of 4.5m error in GPS coordinate estimation as the ground-truth itself can incur an average 3m error (standard GPS accuracy). In the right-most (fourth from left) graph in Fig. 4.5 we show the comparison between camera estimated

distance error and camera estimated location error. We observe that the errors, though not exactly matching in a linear fashion, are consistent with GPS errors, however, are at a lower value.

(2) WiFi FTM Ranging Based Localization Assistance with HD and Pseudo-HD

Setup and Methodology: In this setup we have a static HD, a static pseudo-HD, and a DD flying in proximity of the HD and pseudo-HD ranging from 3 meters to 15 meters. The ranging data collected in this experiment are (i) distance between HD to DD collected using WiFiRttScan, (ii) distance between HD to Pseudo-HD collected using WiFiRTTScan, and (ii) distance between DD to Pseudo-HD collected using WiFiNanScan. We use two WiFiFTM RTT devices and an AP, that is, two pixel phones attached on the pseudo-HD, DD, and AP on the HD. Similar to the camera based localization we validate estimated distance, bearing, and location of our generated results.

Results Discussion: In Fig. 4.6 we show the bespoke accuracy values at different distances, i.e., 3, 5, 7, 10, 15 (meters). The distance groups refer to the ground markers that we mentioned earlier while describing our different experiment setups. We adopt the distance group concept to demonstrate exactly how our localization methods performs in different distances. The first-from-left plot shows absolute errors of bearing estimation for all datasets. The median accuracy of estimated bearing over all the distance groups is 8° with spread of (+/-) 9°. The second-from-left plot shows estimated distance accuracy for all distance groups. The median accuracy of estimated distance over all the distance groups is 1m with spread of (+/-)1m. In the third-from-left plot, the median accuracy of estimated

location over all the distance groups is 1m with spread of (+/-)1m. Overall, the WiFi FTM based results indicate that our measurements are in conformity with the standard FTM ranging accuracy of about 1m. We have been able to demonstrate that by using a P2P WiFi FTM modality, we can estimate the GPS coordinates with the same fidelity as the ranging accuracy for state-of-the-art FTM technology.

4.1.7 Conclusion

In this work we designed and evaluated methods for peer-to-peer truly distributed drone-to-drone localization. The notion is to present a GPS redundancy/alternative to low payload smaller drones with common-already existing on-board hardware and processing power. We designed localization methods that estimate distance, bearing, which are used to compute GPS coordinates between HD and the DD by using a (1) camera (optical) and (2) WiFi (radio) ranging. The camera-based method provides location accuracy in median range of 1–3m with a standard deviation of 2m, and the WiFi FTM ranging based method provides location accuracy of 1m with standard deviation of 1m.

4.2 Modelling and Simulation of Drone Swarm Localization

We complete our drone localization endeavor by appending the drone swarm localization modelling. In this work, we design, implement, and evaluate the drone swarm modelling and simulations where our (i) inter-drone communication prototype and (ii) peer-to-peer Camera-Wifi localization serves as preliminary feasibility study and full-fledged outdoor in-field localization respectively. In our swarm localization model a group of location infor-

mation deprived drones (distressed drone) fly randomly and arbitrarily. Another drone with working navigation location perceives the environment and intentionally comes to look for such location information distressed drones and help those achieve near accurate localization. Moreover, we evaluate the swarm model by designing more than 100 simulated experiments with control parameters, average localization accuracy, and an increasing number of DDs.

4.2.1 Novel Contributions

The aspects of modelling this entire swarm localization is listed below.

1. Modelling a distress drone capable of flying randomly and aimlessly in a specified area and continuously seeking localization help.

2. Modelling a group of distress drones to be able to fly and coexist in a specified area by ensuring collision avoidance.

3. Modelling a helper drone that searches for distressed drones, identify them, calculate their near appropriate relative GPS coordinates, and send coordinates to them.

4. Simulating the entire system model mentioned in points 1, 2, and 3 through repeated experiments using integrated parameter sweeping toolkit to understand the drone swarm location behavior based on multiple control parameters.

4.2.2 Introduction to Modelling and Simulation

In this work, an experimental environment of drone swarm localization simulation is carried out in an arbitrary search space. The search space is assumed to be a open sky. This model

is general enough for it to be implemented in different size search space. We have used Netlogo 6.2.2 (104) platform and it's programming tools for simulating our experiment and result generation.

a. Agents: An agent is defined as a computer system located in an environment and able to accomplish autonomous actions in order to reach its aims in that environment. The agent perceives its environment, acts autonomously, interacts to share the aims, constraints, etc., anticipates and reacts with flexibility with its environment and learns from its experiences and adapts to its environment.

b. Model: A model is a mathematical, graphic and computerized representation of the objects and the relations between them in a confined zone of the real world. A model can also be viewed as a simplified representation of a complex reality. To be useful, models must be adapted to their objects and be conveniently studied and validated.

c. Simulation: Systems composed by a large number of individuals submitted to several environmental variations, which interact between them and with their environment, like helping a group of arbitrarily flying lost drones struggling to navigate themselves, where a helper drone must repeatedly scan a perimeter, avoid obstacles, identify help seeking drones, and send navigation information, etc. this is a complex and dynamic system. As a whole, modelling and simulation first of all consist in the designing of a model. It is a way of making explicit the complexity of a system in order to better understand its functioning and to make good decisions. It brings the complex system to experiment without altering it too much, often difficult in real life situations.

4.2.3 Necessity, Challenges, and Solutions: Drone Swarm Localization Modelling

In this part we address fundamental questions regarding the necessity, challenges, and solutions of a drone swarm localization modelling and simulation tool set.

Why do we need/adopt a simulation platform for drone swarm localization?

To ensure effective and laboratory planned experimentation we would need several drone units and a lot of expensive auxiliary hardware. In-field drone experiments are costly, time-consuming, and greatly impacted by drone failure. Among all these constraints the most restrictive matter is flying a group of drones would require a group of proficient pilots. Reproducing any test cases in the outdoor experimentation area is often not possible. On the contrary the simulation environment permits the researcher modelling the lab designed experiments allowing to mimic near-real-world scenarios.

For implementing drone swarm localization basically which type of simulation platform we need? Ans: Agent based modelling.

In our system a group of drones fly randomly and arbitrarily while perceiving the environment, interacting with other entities, and making decisions along the way. Denoting each drone an agent our system can be mirrored through Agent Based Modelling (ABM). An ABM is a computational model for simulating the navigation and interaction of an autonomous agent with intention of achieving certain goals, while understanding the system behavior and outcomes of a series of algorithmic process execution in that complex environment.

Why use Netlogo than other ABM systems? (Comparison of agent based mod-

eling software)

There are several agent base modelling platforms available, functioning as market pricing forecaster, evolving community/system modeler, social and natural sciences modeler, AI modeler, general purpose agent modelling, human AI modeler, etc. The characteristics of the drone swarm localization operation is a controlled natural phenomenon that can be related to any spontaneous operation of search-identify-rescue/help. For example, a crowd evacuation during an emergency, ant colonization, traffic flow control, etc. Therefore, our needs are best served by adopting a social and natural sciences modeler like Netlogo.

We chose Netlogo based on the perks that it provides for its users and programmers. There are very few social/natural sciences modelers that exist in the current time. Namely those are Mason (105), Repast (106), StarLogoT (107), Netlogo, and Swarm (108). Except StarLogoT and Netlogo all the other platform's programming language is either Java or Objective C. Except Netlogo all of them require heavy computational hardware (GPUs). The project turnover time is exponentially higher in Mason, Repast, Swarm because of the programming language constraints. Contrarily, the programming language of StarLogoT and Netlogo are parallel extensions of Logo programming language, the first and most powerful agent-based modelling language.

Now while choosing between StarLogoT we considered the advancements of NetLogo over StarLogoT, StarlogoT requires multiple virtual machines for each agent type, while Netlogo uses a single virtual machine. While it might seem that using multiple virtual machines in parallel might enhance the model execution time, the communication between agents

operating in different virtual machines overall impacts the execution speed negatively.

Netlogo owns more battles over StarLogoT and over all the above-mentioned platforms, that it is open source to the community to contribute, development up to date, ensures extensibilities through APIs, provides integrated parameter sweeping-tool that allows experimentation with models. Moreover, Netlogo in-an-itself a bundle consisting of components like, programming language, compiler, interpreter, interface builder, simulator, syntax highlighter, a graphics engine (both for web and desktop). Also, Netlogo has a huge pre-built model library (109) for providing examples to the programming community. For so many other reasons, that we would discover throughout our model description, among several agent-based natural science modeler we found Netlogo to be the best suited for drone swarm localization.

What is the general mechanism of Netlogo?

Everything in the Netlogo world is a certain kind of agent. Agents are entities that can follow instructions, preserve states, monitor and memorize states, few agents move, few do not. There are four kinds of agents, turtles, patches, links, observer.

The Netlogo world is made up of 2D gird of patches. Patches are square tiles that cover the simulation space. Patches are marked by 2D coordinates with an origin (0.0) at the center of the word. Patche coordinates are integers because there is no slicing of patches. Patches are alive but they do not move. It is turtles that move around the world over the patches. Turtles have coordinates too for denoting their respective positions. While moving over patches turtles can have decimal value coordinates that indicate a partial or a

non-central presence over a patch.

Links connect any two turtles; they don't have any separate location coordinates except they are connecting two turtles in the field. The observer is defined to observe the Netlogo world from outside of the world itself. Observer does not observe passively but it asks turtles and patches to perform actions. At the initial stage there is no turtle in world, observer creates new turtles and patches (110).

Why Netlogo is an efficient platform to simulate our group localization approaches?

The goals of scientific modeling are compromised if programs are long, cryptic, and platform specific. A NetLogo model is less likely to suffer these problems than one written in common. General-purpose languages like Java and C++. NetLogo is its own programming language, embedded in an integrated, interactive modeling environment. The difficulty of programming in Java or C++ isn't due only to the language itself. It's also due to the complication of the environments. When we add in the added complexity of getting the environment to talk to a modeling library or toolkit, the initial barrier for entry for new programmers becomes quite high—even before they start dealing with the difficulties of the languages themselves. In contrast, the NetLogo environment allows a smooth, almost unnoticeable transition from exploring existing models into programming. Besides, a significant facility applicable in our swarm localization modelling is that, NetLogo is usually faster for models with complex code and smaller numbers of agents. Ultimately, the NetLogo engine is single-threaded, and runs in a single virtual machine. At the operating system level, the

NetLogo application is one process, and the NetLogo engine is one thread within that process (104). Therefore, overhead of communicating between virtual machines in agent level interactions is thankfully absent in Netlogo, which ensures faster simulations.

General Description of Swarm Localization Model: What do we model? What are the aspects of our model implementation?

Our model can be segregated into multiple sub-models. Ultimately our goal is to simulate a swarm of drones flying in the sky randomly and aimlessly in absence of location information. Next, another drone with working navigation location intentionally comes to look for such location information deprived drones (distressed drone) and help those achieve near accurate localization. Therefore, the aspects of modelling this entire course can be listed below.

1. Modelling a distress drone capable of flying randomly and aimlessly in a specified area and continuously seeking localization help.

2. Modelling a group of distress drones to be able to fly and coexist in a specified area by ensuring collision avoidance.

3. Modelling a helper drone that searches for distressed drones, identify them, calculate their near appropriate relative GPS coordinates, and send coordinates to them.

4. Simulating the entire system model mentioned in points a, b, and c through repeated experiments using integrated parameter sweeping toolkit to understand the drone swarm location behavior based on multiple control parameters.

4.2.4 Methodology and Design

(1) Neighbor discovery.

Each drone in our system uses an innovative radio-optical fusion for discovering neighboring drones. We demonstrated a working prototype of this system in our baseline work, an ACK-based inter-drone data transfer prototype using two drones described in the section 4.1.3. Neighbor discovery includes steps: (i) Scan – sensing with WiFi distress signal and camera vision, (ii) Identify – uniquely identify DD and pairing.

(2) Swarm Formation and Localization Process.

As the direction of the motion of DDs are random their is no flocking happens among them. Which ideally justifies the absence of their positioning information. DDs are designed to be moved in a certain speed arbitrarily. Next, the HD appears in the experiment area. HD travels to different directions and scans the neighboring space periodically. The scanning range and angle is defined by the range and angle of camera communication and WiFi channel. While scanning on discovery of DD/s those are added to the neighbor nodeset. Next, HD looks for unlocalized DDs in the neighbor nodeset which then are paired with the HD. Once discovered and paired the HD then estimates the distance to the DD. Distance is represented by total number blocks in between the HD and the discovered DD. The HD then assigns a uniformly picked localization error value from the in-field experiment results. The localization error is chosen depending on the distance group, that the DD falls in. The justification of not modelling the actual GPS estimation in the Netlogo model is that, we already validated the both the camera and WiFi localization methods through outdoor in-

field experiments as shown in our P2P DroneLoc work (described in the sections 4.1.4 - 4.1.6). We refrain to measure and estimate real GPS coordinates in this modelling. Our goal of this modelling is to build a tool for understanding key fundamental behaviour of our P2P localization system in a realistic environment with many drone participants. All the DDs discovered during a scan receives localization help before HD moving forward. Once a DD receives the location information from the HD it will switch from the random hovering mode to a regular motion and gradually will move towards the predefined goal-location. HD repeats above process until all DD's are localized.

(3) Model Design and Parameter Details.

Since the network is mobile the HD and DD may change position from one block to any adjacent block. Therefore, the distance between a HD to a DD and the position of a DD may change during the swarm localization period. However, once a DD is localized by a HD then it is ignored in the subsequent localization phases, even if the DD is discovered multiple times during different scanning processes. So, every DD and HDs needs to preserve some state information during the entire time of the algorithm execution. The state information of a DD are (i) a unique ID, (ii) localization status, (iii) localization error. The scanning and the localization process is limited by the camera and WiFi vision distance constraint. The HD follows the distance group ranges while assigning the localization error value to DDs. For example, the HD scan 8-neighboring blocks to cover a 360° view WiFi localization mode, or the immediately adjacent 3 front blocks while in camera localization mode with FOV approximately 180°. Depending on the FOV of the camera HD scans the next adjacent

5 blocks, or next-next adjacent 7 blocks. The consecutive, 3-5-7 block rows form an angular frontal FOV of the camera. We introduce different fashions and approaches for scanning the search space. For example, (i) changing the number of DDs, (ii) different scanning modes i.e., scanning with only camera, scanning with only WiFi.

4.2.5 Implementation of Drone Swarm Localization Model

4.2.5.1 Implementation and Algorithmic Overview

(1) Preface: Building the Netlogo model Any Netlogo model mandates to override an initialization phase and an execution phase. The initialization phase is implemented by overriding the built in 'setup' procedure, contrarily the execution phase is implemented by overriding the built in 'go' procedure. Agents depending on the modelling needs can be derived by overriding the base agent types. Custom typed agents can preserve states, features, and are allowed to define behaviors. Our Drone swarm localization model contains two different custom typed agents, DD, and HD. Each agent is created with initial states and behavior. For example, a DD initially carries localization_status=FALSE which then changes during the simulation process. Contrarily an HD carries a list of identified DDs. The agents depict their behavior through their actions. For instance, during the simulation the typical DD actions are, setting random head direction, moving to the adjacent ground, seeking help, collision avoidance, etc. We show the model overview in the Fig. 4.7.

(2) Implementation and Algorithmic Flow There are three main components we model in the Drone swarm localization, DD, HD, and the repeated experimentation of the complete

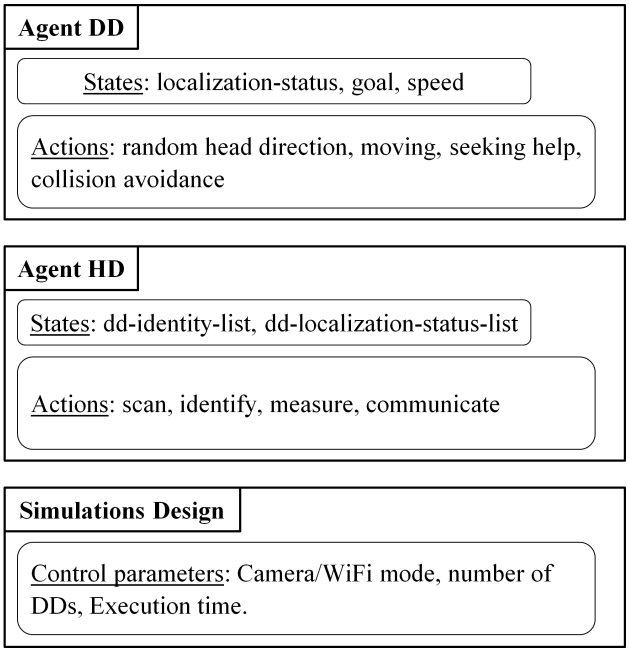


Figure 4.7: Overview of the Drone Swarm Localization demonstrating the model components with their respective states and actions.

drone swarm localization model using parameter-sweeping. Initially we write the setup procedure. An experimentation area is defined by specific number of patches making a 2D grid. Please note that, in our localization algorithms the altitude difference between drones in a certain range does not affect the localization results significantly. Therefore, in our simulations the altitude difference is set to uniform and our simulation ground is set to be a 2D space. We load the data and necessary environment settings in the setup phase too. Then the search space is initiated with a group of DD agents. The DD are landed in the patch grid in a random manner. At the outset, there is no HD present inside the experiment area. Next, it's time to write the 'go' procedure. Beginning of the execution of 'go' the model execution starts. Fig. 4.8 is the flow chart of the drone swarm model.

Upon start of the execution the DD starts hovering the patch grid in different arbitrary

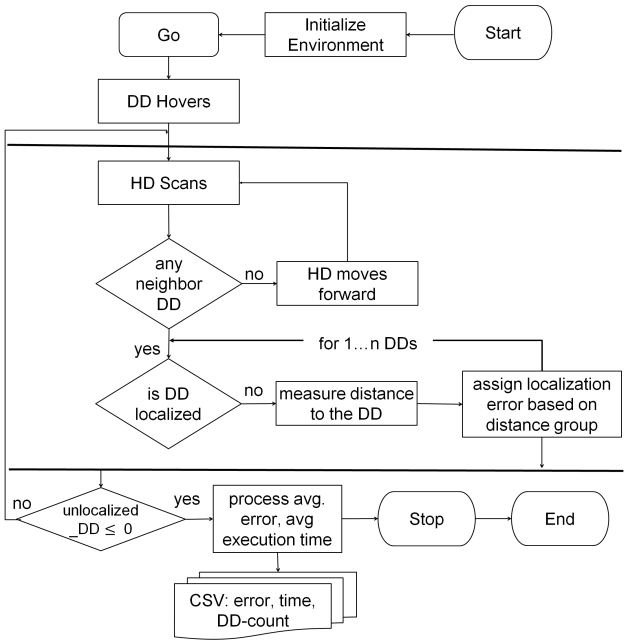


Figure 4.8: The flow chart of the Drone Swarm Localization Model

direction. Hovering is implemented in three main actions, (i) setting a random heading, (ii) perceiving surroundings to ensure collision avoidance, speed control, and maintaining hovering path inside the experimentation space, (iii) moving towards the next patch according to the heading. Hovering continues until the DD gets localization help from a HD.

Next, the HD is launched in the experimentation area by clicking a button from the simulation interface. Once launched the HD lands on an arbitrary patch in the grid. The HD starts scanning the perimeter in a random fashion. HD has the similar behavior implemented like the DD in terms of navigation itself in the experimentation area, i.e, perceiving the environment, speed control, maintain a scanning path, moving and flying around the search space. Significantly, while perceiving the surroundings the HD scans the neighboring region to identify location help seeking lost distressed drones. The scanning process is mentioned already in the methodology and design section. The identification is described in our baseline

work section. Upon identification of one of more DDs the HD measures their distance from itself. HD then pick up a location accuracy data in a uniformly random manner from our previously shown in-field experimentation results according to the distance group that falls under the distance range between the current subject agent-DD and agent-HD. Once all the adjacent discovered DDs receives localization accuracy data the HD moves forward to scan other part of the search area, also the DD's localization status gets updated to TRUE. The stopping condition is maintained by the netlogo-observer, that is, if there is no localization_status=FALSE then it means the HD has done its job and all DDs received localization help. Then the model execution stops. The average localization error for all the DDs and the execution elapsed time is recorded in the result CSV. The Fig. 4.9 shows an ongoing execution screenshot of the drone swarm localization model.

4.2.6 Simulation Design and Evaluation

4.2.6.1 Control and Output parameters

The primary control parameters are the number of drones and the localization mode. Depending on the variable number of drones and scanning mode the required execution time varies significantly. The model execution time also differs by the scanning behavior of the HD and hovering motion of the DDs. We model two localization modes (i) Camera localization, (ii) WiFi Localization. The algorithmic development and in-field evaluation of these two localization methods are described in the 4.1.5 and 4.1.6 sections. In the camera localization mode, HD's vision is determined by the camera vision angle and range. In

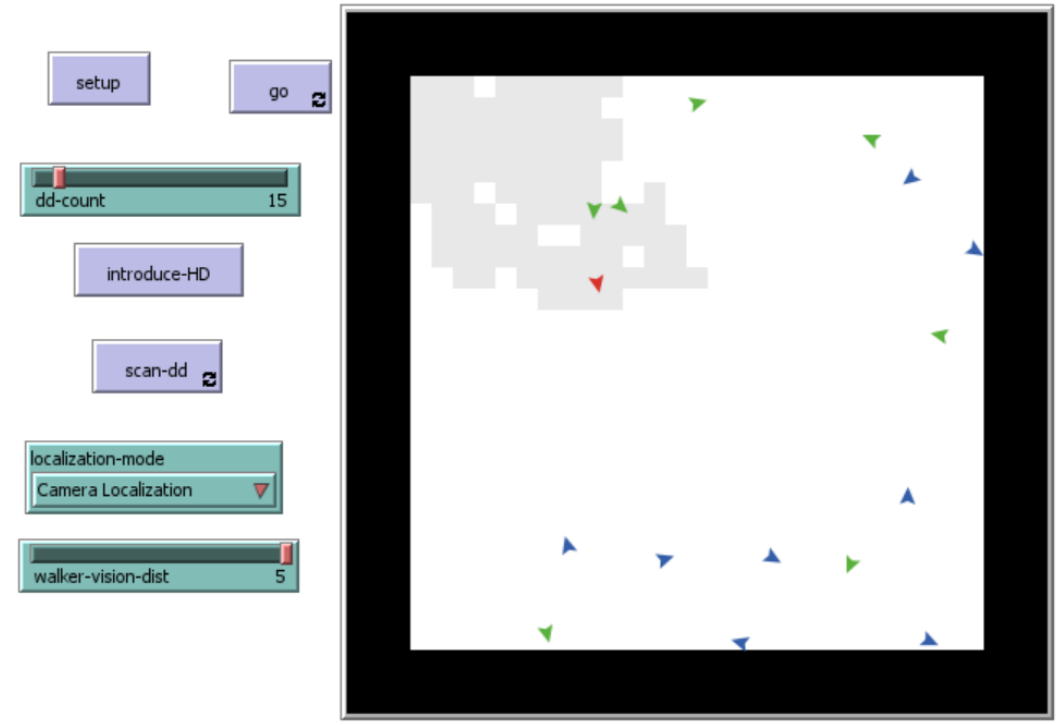


Figure 4.9: A snapshot of the Drone Swarm Localization model while in execution. The arrows are turtles thus agents. Red turtle is the HD, the blue turtles are DDs yet to be localized, the green turtles are DDs received localization help already from the HD. The visited path of the HD is realtime updated and is marked as gray color patches.

the WiFi localization mode HD's vision is defined by the WiFi data propagation angle and range. The output parameters on the other hand are average localization error and time required for model execution completion. The average localization accuracy is measured by the mean localization error over the total number of DDs on completion of a single cycle model execution period.

4.2.6.2 Timing Analysis

Understanding of the time estimation process is necessary before we look into the time evaluation graphs. A single localization operation completion mainly depends on the scanning-

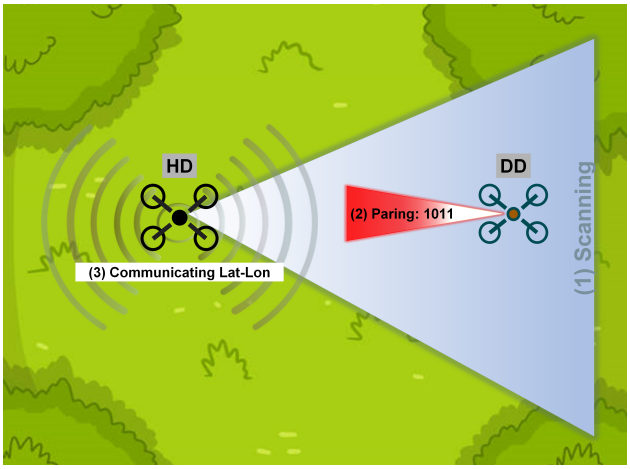


Figure 4.10: A conceptual demonstration of HD operating in the search space starting with travelling and scanning for DDs, then pairing with the discovered DDs, finally estimate and communicate DD's absolute lat-lon.

pairing-communication process. The steps include, (i) HD scanning the surrounding with its sensor (camera/WiFi), (ii) pairing with the detected DD through back-to-back ID communication. (iii) Lat-Lon estimation and communication over WiFi channel. As depicted in the conceptual diagram 4.10.

Scanning includes (a) HD moving forward with a pre-specified speed, and (b) drone detection from the captured image frames. To cover a search area the drone detection is performed in an interval of camera vision range. The camera prototype is capable of perceiving upto 15 meter. Therefore, in every 15 meters the HD performs a camera object detection. The Pixhawk drone used in our prototype maintains a speed of about 10 meter/sec, also the drone detection from a camera image takes around 0.32 seconds (102; 2). For instance, to cover a 40 meter path with the approximately 180° FOV the HD would take about $(40/10) = 4$ sec to travel the path. While travelling this 40 meters HD would attempt for DD detection $(40/15) \cong 3$ times. DD detection attempts take total $(0.32 \times 3) \cong 1$ sec. Therefore, the total

scanning time *(HD travel time + total drone detection time)* = (4 + 1) $\cong 5sec.$

Next, pairing includes both WiFi and led-camera communication. Pairing is described in detail in the Baseline System Design section 4.1.3. The WiFi communication takes about 20ms between drones. The camera-led communication speed is defined by the camera frame rate. In our previous works, drone communication evaluation (102) and mobile VLC demodulation prototype (2) we have demonstrated working camera-LED communication supporting upto 50 bit packets per 1 sec. The Nyquest theory allows a 50b/s sampling in a 100 FPS camera. According to our baseline work for a 4 bits pairing ID we can work with as low as 30 FPS camera. Therefore, each single pairing include *(LED-camera Communication time + WiFi Communication time)*, roughly around (1 + 0.02) = 1.02 sec.

The next part is HD communicating the estimated lat-Lon over WiFi channel. The estimation of the lat-lon takes about 20 ms on the HD side. Then it takes another 20 ms for the DD to receive the lat-lon transmitted by the HD. The estimation and WiFi communication time both are the worst case timing limit. Mostly, the estimation takes about 10 ms and WiFi communication takes about 5 to 10 ms. The Fig. 4.11 demonstrate the entire model execution delay including intermediate components.

4.2.6.3 Timing Calculation in Netlogo Model

We relate the scanning-pairing-communication process in the netlogo world by mapping it's patch and agent behaviour. Since patch grid represents the search space thus each patch represents a unit distance. In our model a single block represents 3-meter real world distance. Three meters is the lowest distance we consider between any HD-DD pair. In accordance

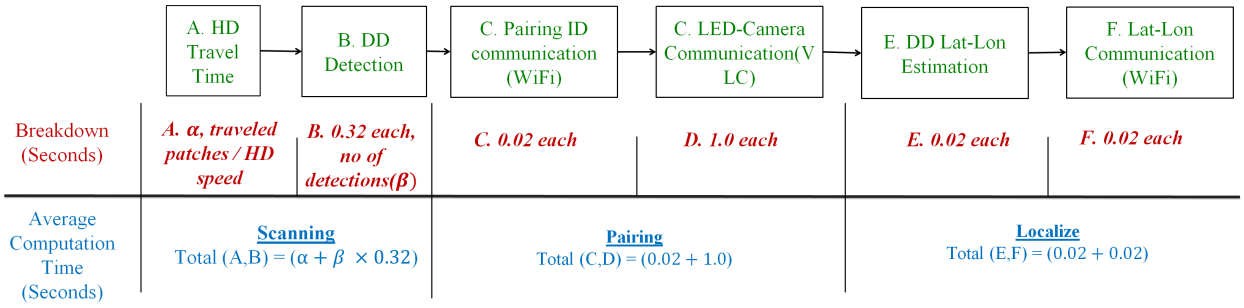


Figure 4.11: Timing Analysis: Drone Swarm Localization Model execution delay including intermediate components. This pipeline represents timing measurements for a single cycle of scanning-pairing-communication during model execution.)

to the Netlogo world, HD travel distance is the number of patches the HD travels in a path. On that note, we find HD travel distance in meter by multiplying HD traveled patches by 3. Then the HD travel time, α is calculated from (HD traveled distance / HD speed). Contrarily, a DD detection attempt is taken in every 15 meters, or 5. patches. Therefore, the number of drone detections is derived by (total number of HD traveled patches/5) = β. The total drone detection time (β * 0.32) sec. Finally, the scanning time is derived from $(\alpha + \beta \times 0.32)$.

Now, for pairing time estimation, a single cycle LED-Camera communication time is *1 sec* and the WiFi communication time is around *20 millisecond*, therefore, a single pairing time $(1 + 0.02)sec$. We recall that HD and DD pairs only when the HD finds an yet to be unlocalized DD. Please note that the pairing count does not exceeds the number of DDs in the experiment space. However, the pairing count may be less or equal to the number of DDs as we allows simultaneous drone discovery. Be the pairing count γ, we estimate the total pairing time during complete model execution time is, $\gamma \times (LED\_camera\_communication\_time + WiFi\_communication\_time)$.

Next the HD estimates DD's lat-lon and the communicate to the DD. A single cycle lat-lon estimation and communication takes about 20 ms on the the HD processing unit, then the lat-lon is communicated in tentatively another 20ms. So, a single estimation and communication consumes $(0.02 + 0.02)$ sec. Moreover, during the entire model execution time the estimation and communiocation will only happens when the HD and DD is paired. So we calculate the total estimation and communication time re-using the pairing count γ. The total estimation-communication time during complete model execution time is, $\gamma \times (0.02 + 0.02) sec.$

4.2.6.4 Result Discussion

In the Fig. 4.12 we present the evaluation of the drone swarm localization model. The experiments are executed for four different DD counts, five, ten, fifteen, and twenty. The complete localization model runs several times repeatedly by parameter sweeping on the control parameters, localization mode [Camera, WiFi], number of DDs [5, 10, 15, 20]. Therefore, we have ζ = (number of DDs * localization mode) number of combinations. However, the model is equipped to allow increase and decrease in DD counts. Primarily, we run our model ζ times to inspect the model behavior in terms of average localization error. Moreover, we then run each combination of parameters several times more to inspect the model behavior in depth.

In the fig 4.12 from left (first) the plot shows the dispersion of average localization error for different dd-counts. We see that the average error value is on the lower side when there are only five DDs in the experiment area. Whereas when the DD-count is higher than the error is much higher. The dd-count vs average error graph shows a lower average error with

a lower number of dds in the experiment area. While the localization error increases with the increase in number of DDs in the experiment area. Possibilities are there were more DDs near the HD and the model's field was crowded. The result analysis from the infield experiments shows that the localization error increases with lower distant DDs. Thus, the error increases with a higher number of DDs being present very close to the HD in the simulation field. Moreover, from our P2P DroneLoc in-field evaluation results it is evident that camera localization error is on the higher side comparative to the WiFi localization.

The 2nd graph from left represents average localization error corresponding to DD distancing (in meters) [3m, 5m, 7m, 10m, 15m]. The DD distancing refers to the different distance groups in which range the HD identified one or more DDs. For instance a DD falls under distance group 7m when the DD and HD was approximately 7 meters apart while localization was conducted. For better understanding the nature of localization accuracy for drones operating in different ranges, and available localization mode (Camera, WiFi) we measure and evaluate our methods in distance groups. This plot likewise shows that the camera localization error is higher. Besides we see that in the both camera and WiFi localization mode the error increases in the 3, 5 meter distance range. This error exaggeration occurs mainly for the inherent behaviour of weak depth estimation performance in the lower range camera projection theory, and lower range congestion possibility in WiFi communication channel.

The 3rd graph from left shows the simulation execution time for different number of DDs present in the search space. The timing is coherently increasing on increase of drone count.

However, depending on the random travelling behaviour of the HD and DDs the simulation conclusion time may differ irrespective to the number of DDs present in the field. As we demonstrated that a unit camera localization takes more time considering the DD detection in the camera image, moreover, the delay increases in camera localization mode because the scanning vision angle in WiFi is 360° whereas, in camera the vision angle is nearly 180°. So, for these reasons compared to WiFi localization mode, it takes more time for the camera localization to be concluded. Besides this timing box plot, we have also presented the average localization error, and model execution times in the tabular form in the Table 4.1. This tabular presents real values from the simulation results to provide the reader a more comprehensive view on the average localization error values and timing segregation in different parts of the simulation process.

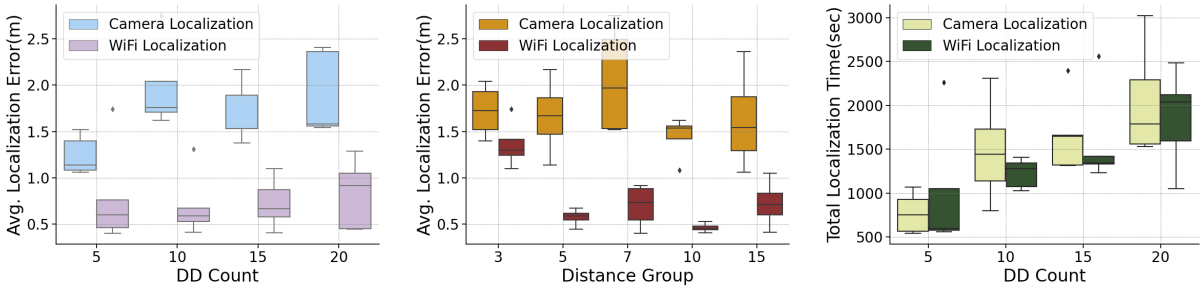


Figure 4.12: Drone swarm localization model simulation results using parameter sweeping. From left, (i) average localization error in relation to DD-count, (ii) average localization error in relation to distance groups, (iii) model execution time in relation to different DD-counts

4.2.7 Conclusion

We design and build a drone swarm localization model using Netlogo programming language and interface. Validate the model through simulations using parameter sweeping with

	DD Count	Avg. Localization Error (m)	Avg. Scanning Time (s)	Avg. Pairing Time (s)	Avg. Communicating Time (s)	Model Execution Time (s)
Camera Localization	5	1.06	263.2	300.1	0.2	563.5
Camera Localization	10	1.71	256.6	540.2	0.4	797.2
Camera Localization	15	1.4	533.6	780.3	0.5	1314.4
Camera Localization	20	2.4	331.6	1200.4	0.8	1532.8
WiFi Localization	5	0.8	257.3	300.1	0.2	557.6
WiFi Localization	10	0.4	741.1	600.2	0.4	1341.7
WiFi Localization	15	0.6	330.8	900.3	0.6	1231.7
WiFi Localization	20	1.1	88.8	960.3	0.6	1049.7

Table 4.1: This tabular holistically represents the average localization error, and model execution time in relation to the localization mode, number of DDs in the field. The distance between HD and DD is within 15 meters. The Model Execution Time is the summation of Avg. Scanning time, Avg. Pairing Time, and Avg. Communicating time. Each parameter combination was repeated four times. For 2 localization modes, 4 different DD-counts, and 4 repetitions, the total number of simulation iterations this tabular presents is $(2 \times 4 \times 4) = 32$. All times are in seconds and error is in meter.

control parameters like average localization error, swarm of DDs. We present an in-depth validation of our previously innovated (i) camera localization, (ii) WiFi localization methods. Conducting around 160 simulated experiments with different control parameter values and combination of repetitions we manifested that our drone swarm localization simulation results are in harmony with the in-field camera and WiFi experiment results. Our Model allows to curate a real-life coordinated drone swarm localization environment setup, provides flexibility to design several simulations with parameter sweeping for significant control parameters like any real-world drone swarm setup would require and leverage. We confirm through our result evaluation that, our innovative drone coordinated camera and WiFi

localization protocols are not only universally usable, also establish a pioneering research opportunity in the drone localization.

CHAPTER 5
CONCLUSION

This thesis presented a novel architecture for NLOS perception for road vehicles, that enables a virtual see through an occluded vehicle functionality. Our system uses a dashboard stereo-camera to perceive the scene in front and communicate that through visible light communication to a follower vehicle. We have designed and implemented a proof-of-concept prototype of a NLOS perception system, that can enable identification of nine safety events, corresponding to traffic light statuses (red, yellow or green), other vehicle merge behaviors, and pedestrian presence. Through experimental evaluations on real-world driving camera footage and real-time trace-based testing, we demonstrated that our system is able to identify occluded events up to 90% accuracy and sustain communication packet error rates in less than 7%. The key usecase of our system is for driving assists, by notifying about safety and time critical events which falls under the level-2 autonomy definition for autonomous vehicles. Such a feature can be very helpful when integrated into a fully autonomous driving vehicle, which plants seeds for the future work emanating from this research. We further extended our NLOS perception work and position a Cognitive Information Processing Pipeline for Multiple–Access Vehicular Camera Communication. We designed an algorithm comprising of prediction attention concepts. In the prediction module we estimate all event occurrence possibilities based on the locally perceived scene data (LOS information). The attention module prioritizes among the probable events and select the most critical one. In addition, we designed a decision-making module that prioritized and chooses the most critical safety

event based on the information from neighboring vehicles. Next, we presented a mobility characterization study through extensive experiments in real world driving scenarios. We characterized motion using a constantly illuminated transmitter on a lead vehicle and a multi–camera setup on a following vehicle. The observations from our experiments reveal key insights on the degree of relative motion of a vehicle along its spatial axis and different vehicular motion behaviors.

With the vehicle-vehicle camera optical communication experiences we extended our quest to address the GPS unreliability problem in drones. We proposed an alternative solution to GPS redundancy for multi drone systems. We presented, implemented and evaluated a design where drones with no GPS or non-functional GPS units are able to receive help to localize themselves coordinated by a helper drone with a functional GPS/location sensing unit. The key concept of our methods is a *helper drone* (HD) tracks nearby GPS *distressed drones* (DD), and then estimates the relative position of the distress drone – by estimating the distance and bearing angle. In our preliminary baseline, we implemented and prototyped an acknowledgement based peer-to-peer (P2P) WiFi communication to conduct inter-drone data transfer between two drones. Next, in our P2P-DroneLoc work, we evaluated two fundamental approaches for range estimation for peer-to-peer relative positioning between drones: (a) camera and computer vision projection theory, and (b) WiFi Fine Time Measurements (FTM). We evaluate our proposed peer-to-peer localization via error across three estimated measures: (i) range or distance between the drones, (ii) GPS bearing (angle), and (iii) GPS location (coordinates). Finally, through design and implementation of a drone

swarm localization model we complete our drone localization endeavor. We designed model simulations using parameter sweeping with control parameters like average localization error, swarm of DDs. We presented an in-depth validation of (i) camera localization, and (ii) WiFi localization methods presented in the P2P-DroneLoc. Conducting around 160 simulated experiments with different control parameter values and combination of repetitions we confirmed that our drone swarm localization simulation results are in harmony with the in-field camera and WiFi experiment results.

The future possibilities and use cases of our research are endless. The intelligent NLOS vehicular perception module can be installed in any system that demands a mobile or even immobile machine intelligence for understanding road or driving situations. By inheriting the proposed concept of our virtual see through work, various multi-modal monitoring systems can use such perception capabilities beyond occlusion. Our drone coordinated localization in P2P and in the swarm mode can be useful for addressing challenges in building practical drone swarm systems.

REFERENCES

[1] K. Ashraf, S. M. T. Islam, A. S. Hosseini, and A. Ashok. Motion characterization for vehicular visible light communications. In 2019 11th International Conference on Communication Systems Networks (COMSNETS), pages 759–764, 2019.

[2] K. Ashraf, V. Varadarajan, R. J. Walden, M. R. Rahman, and A. Ashok. See-through a vehicle: Augmenting road safety information using visual perception and camera communication in vehicles. IEEE Transactions on Vehicular Technology, pages 1–1, 2021.

[3] Ashwin Ashok; Khadija Ashraf; Vignesh Varadrajan; MD Rashed Rahman; Ryan Walden. Non line-of-sight perception vehicular camera communication, 2021.

[4] Distracted driving — nhtsa. `https://www.nhtsa.gov/risky-driving/distracted-driving`, 2018. [Online; accessed 17-July-2020].

[5] Speeding — nhtsa. `https://www.nhtsa.gov/risky-driving/speeding`, 2018. [Online; accessed 17-July-2020].

[6] National motor vehicle crash causation survey. `https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811059`, 2008. [Online; accessed 3-March-2020].

[7] Hang Qiu, Fawad Ahmad, Fan Bai, Marco Gruteser, and Ramesh Govindan. Avr: Augmented vehicular reality. In Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '18, page 81–95, New York, NY, USA, 2018. Association for Computing Machinery.

[8] Richard Hartley and Andrew Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, USA, 2000.

[9] Ashwin Ashok, Marco Gruteser, Narayan Mandayam, Jayant Silva, Michael Varga, and Kristin Dana. Challenge: Mobile optical networks through visual mimo. In Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking, MobiCom '10, page 105–112, New York, NY, USA, 2010. Association for Computing Machinery.

[10] Bugra Turan and Seyhan Ucar. Vehicular visible light communications. Visible Light Communications, page 133, 2017.

[11] A. M. Cailean, B. Cagneau, L. Chassagne, V. Popa, and M. Dimian. A survey on the usage of DSRC and VLC in communication-based vehicle safety applications. In IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT), pages 69–74, November 2014.

[12] T. Komine and M. Nakagawa. Fundamental analysis for visible-light communication system using LED lights. IEEE Transactions on Consumer Electronics, 50(1):100–107, February 2004.

[13] P. Luo, Z. Ghassemlooy, H. Le Minh, E. Bentley, A. Burton, and X. Tang. Fundamental analysis of a car to car visible light communication system. In International Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP), pages 1011–1016, July 2014.

[14] Byung Wook Kim and Sung-Yoon Jung. Vehicle positioning scheme using v2v and v2i

visible light communications. In Vehicular Technology Conference (VTC Spring), 2016 IEEE 83rd, pages 1–5. IEEE, 2016.

[15] Pengfei Luo, Zabih Ghassemlooy, Hoa Le Minh, Edward Bentley, Andrew Burton, and Xuan Tang. Performance analysis of a car-to-car visible light communication system. Applied Optics, 54(7):1696–1706, March 2015.

[16] Nam Tuan Le, Mohammad Arif Hossain, and Yeong Min Jang. A survey of design and implementation for optical camera communication. Signal Processing: Image Communication, 53:95–109, 2017.

[17] Isamu Takai, Tomohisa Harada, Michinori Andoh, Keita Yasutomi, Keiichiro Kagawa, and Shoji Kawahito. Optical vehicle-to-vehicle communication system using led transmitter and camera receiver. IEEE Photonics Journal, 6(5):1–14, 2014.

[18] Yuki Goto, Isamu Takai, Takaya Yamazato, Hiraku Okada, Toshiaki Fujii, Shoji Kawahito, Shintaro Arai, Tomohiro Yendo, and Koji Kamakura. A new automotive vlc system using optical communication image sensor. IEEE photonics journal, 8(3):1–17, 2016.

[19] P. Ji, H. Tsai, C. Wang, and F. Liu. Vehicular visible light communications with led taillight and rolling shutter camera. In 2014 IEEE 79th Vehicular Technology Conference (VTC Spring), pages 1–6, May 2014.

[20] Sung Yooun Jin, Dongnyeok Choi, and Byung Wook Kim. Optical vehicle to vehicle communications for autonomous mirrorless cars. Journal of Multimedia Information System, 5(2):105–110, 2018.

[21] E. Eso, Z. Ghassemlooy, S. Zvanovec, A. Gholami, A. Burton, N. B. Hassan, and O. I. Younus. Experimental demonstration of vehicle to road side infrastructure visible light communications. In 2019 2nd West Asian Colloquium on Optical Wireless Communications (WACOWC), pages 85–89, April 2019.

[22] W. Shen and H. Tsai. Testing vehicle-to-vehicle visible light communications in real-world driving scenarios. In 2017 IEEE Vehicular Networking Conference (VNC), pages 187–194, Nov 2017.

[23] Navin Kumar. Visible light communication based traffic information broadcasting systems. International Journal of Future Computer and Communication, 3(1):26, 2014.

[24] Shlomi Arnon. Optimised optical wireless car-to-traffic-light communication. Transactions on Emerging Telecommunications Technologies, 25(6):660–665, 2014.

[25] Xiaodi You, Yanjun Zhong, Jian Chen, and Changyuan Yu. Mobile channel estimation based on decision feedback in vehicle-to-infrastructure visible light communication systems. Optics Communications, 462:125261, 2020.

[26] Trong-Hop Do and Myungsik Yoo. An in-depth survey of visible light communication based positioning systems. Sensors, 16(5):678, 2016.

[27] V. T. B. Tram and M. Yoo. Vehicle-to-vehicle distance estimation using a low-resolution camera based on visible light communications. IEEE Access, 6:4521–4527, 2018.

[28] Ravi Kumar Satzoda and Mohan Manubhai Trivedi. Looking at vehicles in the night: Detection and dynamics of rear lights. IEEE Transactions on Intelligent Transportation

Systems, 2016.

[29] Trong-Hop Do and Myungsik Yoo. A multi-feature led bit detection algorithm in vehicular optical camera communication. IEEE Access, 7:95797–95811, 2019.

[30] Bastien BÃ©chadergue, Wen-Hsuan Shen, and Hsin-Mu Tsai. Comparison of ofdm and ook modulations for vehicle-to-vehicle visible light communication in real-world driving scenarios. Ad Hoc Networks, 94:101944, 2019.

[31] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. arXiv, 2018.

[32] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR, abs/1704.04861, 2017.

[33] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. CoRR, abs/1506.01497, 2015.

[34] qqwweee. keras-yolo3. `https://github.com/qqwweee/keras-yolo3`, 2018.

[35] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. ArXiv, abs/1704.04861, 2017.

[36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.

[37] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. ArXiv, abs/1704.04861, 2017.

[38] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39:1137–1149, 2015.

[39] Zhiyong Cui, Shao-Wen Yang, and Hsin-Mu Tsai. A vision-based hierarchical framework for autonomous front-vehicle taillights detection and signal recognition. In Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on, pages 931–937. IEEE, 2015.

[40] Khadija Ashraf, S. M. Towhidul Islam, Akram Sadat Hosseini, and Ashwin Ashok. Motion characterization for vehicular visible light communications. In 11th International Conference on Communication Systems & Networks, COMSNETS 2019, Bengaluru, India, January 7-11, 2019, pages 759–764. IEEE, 2019.

[41] What is json. `https://developers.squarespace.com/what-is-json`. [Online; accessed 17-July-2020].

[42] Benjamin Wolfe, Lex Fridman, Anna Kosovicheva, Bobbie Seppelt, Bruce Mehler, Bryan Reimer, and Ruth Rosenholtz. Predicting road scenes from brief views of driving video. Journal of Vision, 19(5):8–8, 05 2019.

[43] Robert G. Gallager. Principles of Digital Communication. Cambridge University Press, 2008.

[44] Kai-Uwe Schmidt and Jürgen Willms. Barker sequences of odd length, 2015.

[45] Yan Zhao and Jayakorn Vongkulbhisal. Design of visible light communication receiver for on-off keying modulation by adaptive minimum-voltage cancelation. Engineering

Journal, 17:125–130, 10 2013.

[46] JD Bullough, K. Sweater Hickcox, TR Klein, and N. Narendran. Effects of flicker characteristics from solid-state lighting on detection, acceptability and comfort. Lighting Research & Technology, 43(3):337–348, 2011.

[47] microsoft. Vott. https://github.com/microsoft/VoTT, 2019.

[48] Richard Szeliski. Computer Vision: Algorithms and Applications. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

[49] Czc auto 12v led magnetic towing light kit. https://www.amazon.com/gp/product/B06ZXWZNMG/ref=ppx_yo_dt_b_asin_title_o03_s00?ie=UTF8&psc=1. [Online; accessed 24-November-2020].

[50] Zed stereo camera. https://www.stereolabs.com/zed/. [Online; accessed 03-August-2020].

[51] Jetson agx xavier. https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit. [Online; accessed 03-August-2020].

[52] Led-stop lamp. https://www.raneystruckparts.com/red-30-led-4-round-stt-competition-series-light/. [Online; accessed 03-August-2020].

[53] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), pages 265–283, 2016.

[54] G. Bradski. The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000.

[55] J. B. Kenney. Dedicated short-range communications (dsrc) standards in the united states. Proceedings of the IEEE, 99(7):1162–1182, 2011.

[56] S. Lee, E. C. Olsen, and W. Wierwille. A comprehensive examination of naturalistic lane-changes. 2004.

[57] Tomer Toledo and David Zohar. Modeling duration of lane changes. Transportation Research Record, 1999(1):71–78, 2007.

[58] Guidelines for timing yellow and all-red intervals at signalized intersections. `http://redlightrobber.com/red/links_pdf/NCHRP-Guidelines-for-Timing-RPT-731.pdf`, 2012. [Online; accessed 24-November-2020].

[59] C. Tebruegge, A. Memedi, and F. Dressler. Reduced multiuser-interference for vehicular vlc using sdma and matrix headlights. In 2019 IEEE Global Communications Conference (GLOBECOM), pages 1–6, 2019.

[60] S. Chen and C. Chow. Color-shift keying and code-division multiple-access transmission for rgb-led visible light communications using mobile phone camera. IEEE Photonics Journal, 6(6):1–6, 2014.

[61] T. Kondo, Ryotaro Kitaoka, and W. Chujo. Multiple-access capability of led visible light communication with low-frame-rate cmos camera for control and data transmission of mobile objects. 2015 IEEE/SICE International Symposium on System Integration (SII), pages 678–683, 2015.

[62] Fan Yang, Shining Li, Zhe Yang, Cheng Qian, and Tao Gu. Spatial multiplexing

for non-line-of-sight light-to-camera communications. IEEE Transactions on Mobile Computing, 18(11):2660–2671, Nov 2019.

[63] Jungyeol Kim, Saswati Sarkar, Santosh S. Venkatesh, Megan Smirti Ryerson, and David Starobinski. Modeling information propagation in general v2v-enabled transportation networks, 2019.

[64] Xiaoxu Liu, Minh Neuyen, and Wei Qi Yan. Vehicle-related scene understanding using deep learning. In Michael Cree, Fay Huang, Junsong Yuan, and Wei Qi Yan, editors, Pattern Recognition, 2020.

[65] Lionel Heng, Benjamin Choi, Zhaopeng Cui, Marcel Geppert, Sixing Hu, Benson Kuan, Peidong Liu, Rang Nguyen, Ye Chuan Yeo, Andreas Geiger, Gim Hee Lee, Marc Pollefeys, and Torsten Sattler. Project autovision: Localization and 3d scene perception for an autonomous vehicle with a multi-camera system, 2019.

[66] Liangfu Chen, Zeng Yang, Jianjun Ma, and Zheng Luo. Driving scene perception network: Real-time joint detection, depth estimation and semantic segmentation. 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Mar 2018.

[67] Steven Vanmarcke and Johan Wagemans. Rapid gist perception of meaningful real-life scenes: Exploring individual and gender differences in multiple categorization tasks. i-Perception, 6(1):19–37, 2015. PMID: 26034569.

[68] N. Pugeault and R. Bowden. How much of driving is preattentive? IEEE Transactions on Vehicular Technology, 64(12):5424–5438, 2015.

[69] Benjamin Wolfe, Lex Fridman, Anna Kosovicheva, Bobbie Seppelt, Bruce Mehler,

Bryan Reimer, and Ruth Rosenholtz. Predicting road scenes from brief views of driving video. Journal of Vision, 19(5):8–8, 05 2019.

[70] C. Olaverri-Monreal, P. Gomes, R. Fernandes, F. Vieira, and M. Ferreira. The see-through system: A vanet-enabled assistant for overtaking maneuvers. In 2010 IEEE Intelligent Vehicles Symposium, pages 123–128, 2010.

[71] G. Marchetti. Consciousness: A unique way of processing information. Cognitive Processing, 19(3):435–464, 2018.

[72] Cooper A. Smout, Matthew F. Tang, Marta I. Garrido, and Jason B. Mattingley. Attention promotes the neural encoding of prediction errors. bioRxiv, 2019.

[73] C.Y.D. Yang. Analysis of red light violation data collected from intersections equipped with red light photo enforcement cameras. AAA Foundation for Traffic Safety, 2006.

[74] Wen-Hsuan Shen and Hsin-Mu Tsai. Testing vehicle-to-vehicle visible light communications in real-world driving scenarios. In Vehicular Networking Conference (VNC), 2017 IEEE, pages 187–194. IEEE, 2017.

[75] Takaya Yamazato, Masayuki Kinoshita, Shintaro Arai, Eisho Souke, Tomohiro Yendo, Toshiaki Fujii, Koji Kamakura, and Hiraku Okada. Vehicle motion and pixel illumination modeling for image sensor based visible light communication. IEEE Journal on Selected Areas in Communications, 33(9):1793–1805, 2015.

[76] Cen B Liu, Bahareh Sadeghi, and Edward W Knightly. Enabling vehicular visible light communication (V2LC) networks. In Proceedings of the Eighth ACM international workshop on Vehicular inter-networking, pages 41–50, 2011.

[77] Shintaro Arai, Yasutaka Shiraki, Takaya Yamazato, Hiraku Okada, Toshiaki Fujii, and Tomohiro Yendo. Multiple LED arrays acquisition for image-sensor-based I2V-VLC using block matching. In Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th, pages 605–610, 2014.

[78] Isamu Takai, Shinya Ito, Keita Yasutomi, Keiichiro Kagawa, Michinori Andoh, and Shoji Kawahito. LED and CMOS image sensor based optical wireless communication system for automotive applications. IEEE Photonics Journal, 5(5):6801418–6801418, 2013.

[79] Tsubasa Saito, Shinichiro Haruyama, and Masao Nakagawa. A new tracking method using image sensor and photo diode for visible light road-to-vehicle communication. In Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on, volume 1, pages 673–678, 2008.

[80] Masayuki Kinoshita, Takaya Yamazato, Hiraku Okada, Toshiaki Fujii, Shintaro Arai, Tomohiro Yendo, and Koji Kamakura. Motion modeling of mobile transmitter for image sensor based i2v-vlc, v2i-vlc, and v2v-vlc. In Globecom Workshops (GC Wkshps), 2014, pages 450–455. IEEE, 2014.

[81] Hai-Yan Zhang. Multiple moving objects detection and tracking based on optical flow in polar-log images. In Machine Learning and Cybernetics (ICMLC), 2010 International Conference on, volume 3, pages 1577–1582. IEEE, 2010.

[82] Taha Kowsari, Steven S Beauchemin, and Ji Cho. Real-time vehicle detection and tracking using stereo vision and multi-view adaboost. In Intelligent Transportation

Systems (ITSC), 2011 14th International IEEE Conference on, pages 1255–1260. IEEE, 2011.

[83] Donguk Seo, Hansung Park, Kanghyun Jo, Kangik Eom, Sungmin Yang, and Taeho Kim. Omnidirectional stereo vision based vehicle detection and distance measurement for driver assistance system. In Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE, pages 5507–5511. IEEE, 2013.

[84] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A toolbox for easily calibrating omnidirectional cameras. In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 5695–5701. IEEE, 2006.

[85] Berthold Horn, Berthold Klaus, and Paul Horn. Robot vision. MIT press, 1986.

[86] Richard Hartley and Andrew Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[87] Yu Liu, Shuping Liu, Yang Cao, and Zengfu Wang. A practical algorithm for automatic chessboard corner detection. In Image Processing (ICIP), 2014 IEEE International Conference on, pages 3449–3453. IEEE, 2014.

[88] haversine. `https://www.math.ksu.edu/~dbski/writings/haversine.pdf`. (Accessed on 09/07/2018).

[89] Michael Varga, Ashwin Ashok, Marco Gruteser, Narayan Mandayam, Wenjia Yuan, and Kristin Dana. Visual mimo based led-camera communication applied to automobile safety. In Proceedings of the 9th International Conference on Mobile systems, Applications, and Services, pages 383–384, 2011.

[90] R. I. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[91] Lili Chen, Jie Xiong, Xiaojiang Chen, Sunghoon Ivan Lee, Kai Chen, Dianhe Han, Dingyi Fang, Zhanyong Tang, and Zheng Wang. Widesee: Towards wide-area contactless wireless sensing. NY, USA, 2019. Association for Computing Machinery.

[92] Juhwan Noh, Yujin Kwon, Yunmok Son, Hocheol Shin, Dohyun Kim, Jaeyeong Choi, and Yongdae Kim. Tractor beam: Safe-hijacking of consumer drones with adaptive gps spoofing. 22(2), 2019.

[93] Inseop Um, Seongjoon Park, Hyeong Tae Kim, and Hwangnam Kim. Configuring rtk-gps architecture for system redundancy in multi-drone operations. IEEE Access, 2020.

[94] Joshua D. Redding, Timothy W. McLain, Randal W. Beard, and C.N. Taylor. Vision-based target localization from a fixed-wing miniature air vehicle. 2006 American Control Conference, pages 6 pp.–, 2006.

[95] Subong Sohn, Bhoram Lee, Jihoon Kim, and Changdon Kee. Vision-based real-time target localization for single-antenna gps-guided uav. IEEE Transactions on Aerospace and Electronic Systems, 2008.

[96] V.N. Dobrokhodov, I.I. Kaminer, K.D. Jones, and R. Ghabcheloo. Vision-based tracking and motion estimation for moving targets using small uavs. In 2006 American Control Conference, pages 6 pp.–, 2006.

[97] Ross W. Deming and Leonid I. Perlovsky. Concurrent multi-target localization, data

association, and navigation for a swarm of flying sensors. 2007.

[98] Sara Minaeian, Jian Liu, and Young-Jun Son. Vision-based target detection and localization via a team of cooperative uav and ugvs. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 46(7), 2016.

[99] Jan P. Hogendijk. Glen van brummelen. heavenly mathematics: The forgotten art of spherical trigonometry. Isis, 105(1):207–208, 2014.

[100] Anwen Wang, Xiang Ji, Dajun Wu, Xuedong Bai, Nana Ding, Jingming Pang, Shaofeng Chen, Xiaojiang Chen, and Dingyi Fang. Guideloc: Uav-assisted multitarget localization system for disaster rescue. Mob. Inf. Syst., 2017.

[101] Victor Delafontaine, Fabrizio Schiano, Giuseppe Cocco, Alexandru Rusu, and Dario Floreano. Drone-aided localization in lora iot networks. In 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020.

[102] Bhawana Chhaglani, Abhay Sheel Anand, Nakul Garg, and Ashwin Ashok. Evaluating led-camera communication for drones. NY, USA, 2020. Association for Computing Machinery.

[103] Chris Veness. Destination point along great-circle given distance and bearing from start point. `http://www.movable-type.co.uk/scripts/latlong.html`, 2022. [Online; accessed 15-May-2022].

[104] Seth Tisue and Uri Wilensky. Netlogo : Design and implementation of a multi-agent modeling environment. 2004.

[105] Sean Luke, Claudio Cioffi, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A

multiagent simulation environment. Simulation, 81:517–527, 01 2005.

[106] Nicholson Collier, T Howe, Robert Najlis, Michael North, and J. Vos. Repast 3.0: Recursive porous agent simulation toolkit. 01 2007.

[107] Uri Wilensky. Starlogot. evanston, il. center for connected learning and computer based modeling, northwestern university. 1999, 2001, 2002.

[108] Nelson Minar, Roger Burkhart, Chris Langton, and Manor Askenazi. The swarm simulation system: A toolkit for building multi-agent simulations. Santa Fe Institute Working Paper, 96-06-042, 07 1996.

[109] Netlogo models library. `https://ccl.northwestern.edu/netlogo/models/index.cgi`. [Online; accessed 29-Aug-2023].

[110] The beginner's guide to netlogo programming. `https://ccl.northwestern.edu/netlogo/bind/`. [Online; accessed 29-Aug-2023].