Digital Twin Simulation for Traffic Infrastructure Management in Bangladesh

by

Ahmed Awsaf
21201371
Sajida Hossain
24341107
Purba Tahia Hassan
24341105
Zabir Ramiz
24341108
Md. Amir Ul Islam
23341077

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
February 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Zabir Ramiz
24341108

Purba Tahia Hassan
24341105

Ahmed Awsaf
21201371

Md. Amir Ul Islam
23341077

Sajida Hossain
24341107

Approval

The thesis/project titled "Digital Twin Simulation for Traffic Infrastructure Management in Bangladesh " submitted by

1. Ahmed Awsaf (21201371)
2. Sajida Hossain (24341107)
3. Purba Tahia Hassan (24341105)
4. Zabir Ramiz (24341108)
5. Md.Amir Ul Islam (23341077)

Of Fall, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on February 03, 2025

Examining Committee:

Supervisor:
(Member)

Dr. Jannatun Noor Mukta

Associate Professor, CSE
Director, BSDS
United International University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi

Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Bangladesh has been facing tremendous challenges in the traffic sector in the last decade. The capital city Dhaka is one of the most densely populated cities in the world. With the rising number of vehicles and the increasing population, traffic infrastructure development has proven to be a big hurdle to overcome. The need for digital infrastructure has added to the existing challenges of traffic infrastructure management. New technologies like Digital Twins can play a vital role in this scenario. A digital twin is the virtualization of a physical infrastructure. This technology has been adopted in many countries for its cost-effectiveness and versatility. Using a digital twin of the traffic network is now crucial to effectively manage and monitor the traffic infrastructure of Dhaka. This research explores a dynamically simulated traffic environment in a Digital Twin of an area in Dhaka and a customizable platform tailored for traffic infrastructure focused on traffic flow in Bangladesh. The digital twin aims to simulate potential intersection scenarios to showcase granular insights into vehicular traffic data and tries to signify the high-traffic zones. An ML model will be specifically trained to detect traffic data and used as the basis for the traffic simulation. This traffic data paired with behavioral coding of cars will create a realistic traffic simulation with synonymous scenarios. It will target the groundwork of virtualization and turn intersections into simulated desk environments to reduce the need for extensive practical experimentation. It will focus on speculative experiments that can later be implemented in real-world environments to be determined feasible.

Keywords: Digital Twin, Traffic Management, Urban Planning, Simulation, YOLO, SUMO, Unity

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Traffic management has been marked as a major issue in Bangladesh. A statistical analysis [31] shows Bangladesh bleeds USD 2.01 million in traffic congestion per day. The amount derives from a combination of causes, which include the lack of productivity in the workplace due to high travel time, fuel inefficiency, and the delay of transport of perishable goods. It is a huge economic loss for a small developing country like Bangladesh. The daily frustration of spending hours on the road has also taken a toll on the morale and psyche of citizens. Throughout the past decade, the people of Bangladesh have shown those frustrations in the form of protests, artistry, and temporal literature [39]. The cause for this dire situation can be narrowed down to the lack of knowledge, unenforced laws, and proper management of traffic [13]. These problems form complex and unpredictable scenarios. Moreover, Lax traffic law enforcement has confused drivers regarding the proper interpretation of traffic rules in Bangladesh. Irregular traffic patterns, unpredictable driving behavior, and inconsistent adherence to rules combined with improper traffic management have created a chaotic environment that is challenging to manage. Therefore, foreign methods of controlling traffic can be deemed ineffective in managing the unique Bangladesh traffic condition. To tackle the unique case of Bangladesh, we need to find a better solution to the unpredictable nature of traffic.

This research plans to provide a feasible solution to managing Bangladesh's traffic infrastructure. Given the challenges, effective management must be implemented to create sustainable development in Bangladesh. The study done in [15], shows an example of effective urban planning done in Zurich, Switzerland, by implementing a highly detailed digital twin of the city to create an overview of the current infrastructure and pave the way for any future planning on the existing infrastructure. The research proves the effectiveness of using a digital twin in the management, development, maintenance, and innovation of all parts of a city. One sector this can be implemented in is traffic infrastructure management. In countries like China, as shown in [42], digitized city plans paired with AI models were implemented to improve transportation infrastructure. The research highlights how AI models have reduced traffic congestion so that they can evolve into incorporating autonomous vehicles into the traffic flow. It specifically states how digitizing road maps has helped create construction plans and route changes to improve the overall traffic congestion in the city. Our research tries to implement a similar utility to improve how traffic is analyzed relative to the traffic situation in Bangladesh.

This study will attempt to create a digital twin of a high traffic area in Dhaka. More specifically, the busy intersection of Gulshan 2 in the heart of Dhaka has been chosen as the representative intersection for this study. This research plans to create a utility system that maps out a digital twin of the intersection and its surrounding areas. The three-dimensional model of the map will be modified to map out road borders, their respective directions, and traffic signs. Here, simulated objects that represent vehicles of different types while complying with the specified rules will move with coded behavior. The volume of traffic, speed, and lane data will be collected through recorded data to match the traffic volume to the real-life scenario. Overall, the miniature model will show potential choke points on the map by obstructing roads that may result in massive congestion. Choke points in this part can be said to be the point where traffic is obstructed or a point on the intersection that can cause traffic jams. Therefore, the analyst using this system will be able to change traffic props and alter rules to benchmark the current traffic. Through the digital twin, the research will prove effective traffic analysis with accurate simulations of the environment.

1.1 Motivation

The traffic volume-to-capacity ratio is considered a crucial part of any traffic analysis. Bangladesh has one of the highest ratios in the world. Research by the Department of Civil Engineering, Ahsanullah University of Science and Technology [38], shows that during peak hours of a moderate traffic day on the route from Bijoy Sharani to Shatrasta, an average of 1713 PCU/hr is calculated on the 120-foot road in 30 minutes. The unit PCU/hr shows the flow rate/hr of the traffic on that road. This value has been described as an "F" tier level of service by the Engineers of Rajshahi University [5]. This label tells us that at peak hours in Dhaka, the Bijoy Sharani intersection, one of the busiest roads in Dhaka, shows the worst kind of traffic with slow travel times and exceeding the capacity of the road. The capital controls the country's economy, yet its horrible traffic is holding it back from its full potential. Bangladesh needs a proper method of planning in the transportation engineering industry.

This study's impact lies in its ability to offer valuable data on rush hour times, traffic flow patterns, parking zones, and traffic-intensive areas like central business districts and high-traffic buildings tackling the current situation. Our approach includes the development of an interactive simulation framework that processes video footage and translates it into plausible traffic flow information. This data helps the framework enhance its ability to detect traffic behavior such as lane changing, reaction time to signals, and traffic speed. Specifically trained YOLO models detect traffic data such as mentioned and are used as the basis for the traffic simulation. This traffic data paired with behavioral coding of cars will create a realistic traffic simulation with synonymous scenarios.

The benefit of this approach is immense. This study not only gives us an easy-to-use system but also a highly cost-efficient one. For a country like Bangladesh [31], the lack of planning can be labeled as an opportunity cost. While the people are struggling financially, the country would need funding for more important quality of life problems. The resources used will be near zero for every step and already open-source resources are used to keep cost to the minimum.

1.2 Limitations

Considering Bangladesh's increasing traffic issue, especially in urban areas like Dhaka, no research or project till now has effectively implemented a Digital Twin for traffic control. Bangladesh traffic management is inefficient as it does not use modern technology, old traffic infrastructure, or manual traffic control [55]. The following limitations highlight the need for a Digital Twin technology for traffic control as well as the constraints hindering the adoption of such a smart system in Bangladesh.

- **Incompatibility of existing Digital Twin implementations with Bangladesh's traffic infrastructure:**
  While DT technology has shown success in well-organized settings in wealthy nations, its use in unstable and chaotic traffic environments like Bangladesh's is continuously developing. The majority of DT models are less useful from Bangladesh's perspective since they focus on places like Singapore, Beijing, Zurich, and other cities with rigorous traffic laws [62][15].

- **Inefficiencies in Bangladesh's current traffic management system:**
  Bangladesh's existing traffic management system is quite inefficient and has some problems. It is dependent on manual control of traffic, which results in delays, inconsistencies, and accidents caused by people. Additionally, there aren't enough real-time monitoring systems to keep track of traffic patterns, collisions, or congestion, which leads to impulsive rather than preventive traffic management strategies [43]. Disorganized road construction and urban planning frequently result in an imbalance between many vehicles and road capacity, which causes traffic jams, especially in cities like Dhaka [55]. Effective management of traffic is further complicated by the general lack of public knowledge and frequent illegal acts of traffic laws by drivers.

- **Absence of proper Digital Twin infrastructure for Bangladesh's traffic system:**
  Bangladesh is still not utilizing DT technology for traffic control because of several issues. The nation needs the digital infrastructure required to establish and maintain a DT system, such as high-speed communication networks and real-time sensors [15]. Furthermore, it can be difficult to build a strong DT model due to the lack of a consistent infrastructure and dependence on manual data collection in existing approaches, which result in an absence of accurate traffic data. Bangladesh, being a developing nation, suffers from cost limitations that hinder the use of such advanced technologies. Moreover, a large number of technical and financial resources are required for the implementation and maintenance of DT technology.

- **Lack of effective implementation of adopted traffic solutions:**
  Although Bangladesh has adopted numerous traffic management strategies from other countries, many of these have not been successfully implemented because they have not been appropriately transformed to the local environment. In some places, technologies like Intelligent Transport Systems, automated traffic signals, and traffic signal control are in use [43] but because of limited service, uneven enforcement, and a lack of integration with real-time traffic data, these systems do not perform as well as they could.

- **Limited success of traditional traffic control approaches:**
  The success rate of traditional traffic management techniques, such as building flyovers and subways, construction of at least four-lane roads, and extending existing roads [79], in reducing traffic congestion remains inconsistent. These solutions are costly, time-consuming, and sometimes unable to change with the dynamic traffic situations found in places like Dhaka.

Due to the country's heavy reliance on manual traffic control, outdated infrastructure, and a lack of modern technology integration, Bangladesh's traffic management system is still quite ineffective. Long waiting times and financial losses are the results of an overwhelming inability to handle the increasing congestion due to a lack of real-time monitoring, data-driven solutions, and predictive traffic control. These drawbacks emphasize the necessity of more advanced techniques, such as Digital Twin technology, to provide efficient traffic management and infrastructure improvement.

1.3 Research Objectives

The goal of this research is to propose a methodology for a digital twin focusing on the traffic infrastructure of Bangladesh. This research will provide an environment that will simulate realistic scenarios as an insight into the traffic flow that can be utilized in the urban planning process of Bangladesh. The objective of this research is to

- **Develop a traffic infrastructure model:**
  A realistic area map with 3D visuals and physics to simulate the traffic flow of a particular area in Dhaka.

- **Assist in Urban traffic analysis:**
  The environment will give designers a better overview of the traffic to help identify choke points.

- **Develop path-finding and path-selecting objects:**
  A system particularly tailored to the driving behaviors and traffic patterns typical of Bangladeshi drivers.

- **Simulate Real-Time Traffic:**
  The traffic simulations will orchestrate possible scenarios for further improvement and benchmark the current infrastructure's capacity.

- **Predict possible hurdles for improvement:**
  The simulation can show flaws in speculative laws through feasibility analysis of the simulation. This can include checking if stricter international laws can improve the current flow of traffic.

- **Create a Sandbox toolkit:**
  Urban analysts can edit traffic props, rules, lane uses, types, and weights to specify which combination has the optimal output.

1.4 Research Contribution

This research mainly targets the field of Urban Traffic Infrastructure in Bangladesh.

- **Development of a Digital Twin for Traffic Simulation**
  This research presents a practical and innovative method of utilizing simple tools to create a Digital twin of Dhaka city. It introduces an implementation of the Digital Twin of an intersection in Dhaka by presenting the 2D and 3D Digital Twin into an interactive analysis tool that incorporates traffic flow, behaviors, and basic modification properties to visualize a realistic version of the intersection.

- **ML-Based Traffic Data Collection**
  The study integrates an ML-powered approach to extract traffic data by utilizing computer vision techniques, such as YOLO-based object detection. This method enables real-time traffic data extraction. Such Ml-based techniques enhance the accuracy of traffic simulations and predictions.

- **Identification of Traffic Choke Points and Optimization Techniques**
  This study provides an efficient method for locating important choke points in Dhaka city that cause congestion by simulating different traffic scenarios. The suggestion approach allows researchers to evaluate various alterations to traffic signals, infrastructure, and rules in a simulated setting before putting them into practice in real-world scenarios.

- **Cost-effective and Scalable Traffic Planning Framework**
  The implementation of this research is designed to be highly cost-effective, utilizing open-source tools and minimal computational resources. The low cost and lowered implementation mean the toolset can be scaled and adjusted for larger test zones with limited hardware.

- **Policy and Infrastructure Recommendations**
  The research aids in the development of data-driven traffic policies by providing insights regarding traffic flow inefficiencies and potential improvements. The findings can assist urban planners and policymakers in transportation strategies to reduce congestion as well as designing optimized road networks.

- **Bridging the Gap in Smart City Initiatives**
  Digital Twin technology is still not completely included in Bangladesh's smart city plans. This paper provides the groundwork for using modern simulation methods for urban traffic control.

1.5 Thesis Organization

The thesis begins with an abstract that provides a high-level overview of the research. chapter 1 introduces the research problem, motivation, objectives, and intended contributions. chapter 2 reviews relevant literature, while chapter 3 outlines the background, including key terms, datasets, and technologies used. chapter 4 details the work plan and dataset preprocessing. chapter 5 presents the experimental setup and results, followed by chapter 6, which discusses the findings and their interpretations. chapter 7 highlights the limitations encountered and potential future work. Finally, chapter 8 summarizes the key findings and concludes the thesis.

Chapter 2

Literature Review

A comprehensive literature review is essential to understand the current state of research on Digital Twins and their applications in overall urban planning and traffic management. The following review focuses on examining existing studies in order to identify gaps in knowledge and explore how Digital Twin technology can be effectively implemented to handle traffic management in the unique context of Bangladesh.

2.1 Overview of Digital Twin

Grieves and Vickers [4] defined the Digital Twin (DT) as "a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro-atomic level to the macro-geometrical level. At its optimum, any information that could be obtained from inspecting a physically manufactured product can be obtained from its Digital Twin". DT tries to combine the physical and virtual worlds by twinning, simulation, real-time monitoring, analytics, and optimization. It has also been recognized as the next breakthrough in digitization and simulation [1]. The article presents in [49], that the primary aim of DT was recognized to be a digital model of a physical entity that reflects its physical behavior by applying platforms and two-way interactions of data in real-time.

The article in [49] described the categorization of DT in detail. They categorized the fundamental notions that describe the DT technology in three ways:

- **DT prototype (DTP):** It is the model illustration of the physical twin. The virtual drawing of a particular entity. This is often created before the physical product is created [54].
- **DT instance (DTI):** It is the instance created from the DT prototype. It is applied to a manufactured product to conduct different experiments [54].
- **DT aggregate (DTA):** This is a group of DT instances and other DT aggregates. It combines multiple DT instances and aggregates to better understand the environment. It determines the product's capabilities, executes prototypes, and tests operating variables [54].

With the advent of smart cities, DT has received significant attention for improving smart city capabilities. The research in [54], explained the extended categorization

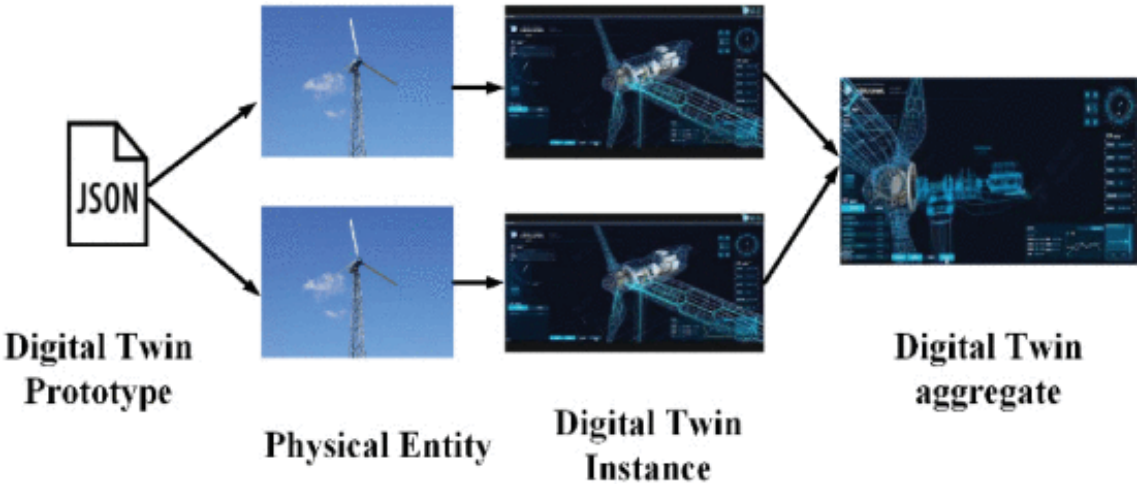

Figure 2.1: Correlation between DT concepts [49]

of DT and focused on the process of integrating the Internet of Things (IoT) to enable various smart city services. The real-time data received from various sensors and IoT technologies continuously optimized the robustness of the DT and provided granular insights into the smart city.

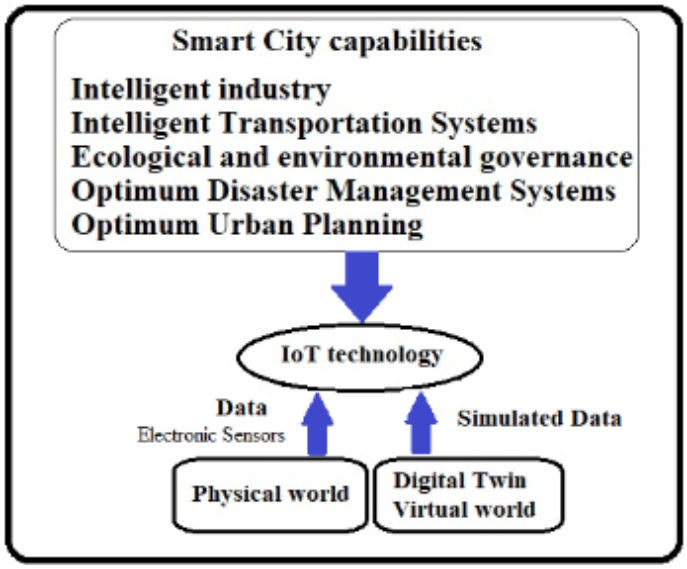

Figure 2.2: Interaction between digital twin's components and smart city [54]

2.2 Digital Twin in Urban Planning

Utilizing Digital Twins in urban planning is an emerging concept that can allow city planners, engineers, and architects to effectively visualize and optimize urban infrastructure and services. Comprehensive studies done in [26][60] show how creating these dynamic, digital replicas of physical urban environments can assist in city planning in several significant ways. These include real-time monitoring using integrated sensors and IoT devices to surveil urban processes such as traffic flow and energy consumption, simulating and testing various real-life scenarios to assess the

impact and make data-driven decisions of potential changes like new transportation systems or responses to natural disasters [60]. Consequently, real-time data integration can enhance situational awareness and enable planners to respond swiftly to any kind of emerging challenge.

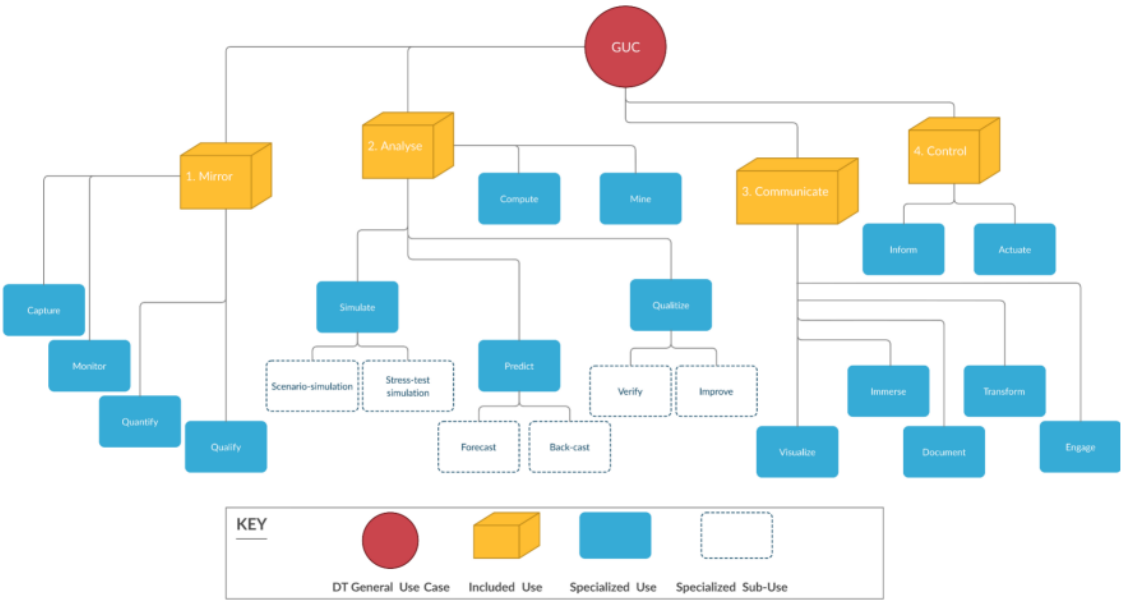

Figure 2.3: Conceptualization of the Digital Twin Uses Classification System [DTUCS] [26]

To better understand and categorize the diverse applications of digital twins, authors in [26] proposed a three-dimensional classification framework based on the urban planning level (strategic, operational, or tactical), planning aspect (environmental, social, or economic), and smart city dimension (people, governance, living, and so on). For example, [14] focuses on the potential applications of digital twins, particularly in the context of urban freight transport. The research uses visualization of complex relationships, such as simulating the effects of placing a microhub in a residential area and showing changes in cargo movements and vehicle types. These are then used to make informed decisions, including strategic long-term planning and policy-making, setting the overall direction for urban freight systems, translating these strategic goals into actionable plans at a tactical level, and day-to-day management of the systems at an operational level, each planning level addressing different aspects of the complex urban logistics systems.

Case studies of several cities in countries around the globe, conducted in [11][15][20][30] further demonstrate how digital twins can be significantly helpful in improving the design and management of urban development initiatives for building smart cities. Their methodology includes the integration of Building Information Modeling (BIM) and Geographic Information Systems (GIS) which allow for a more holistic view of the urban infrastructure, improving the accuracy of urban simulations by facilitating better data management and analysis. These advanced models can then be used to analyze complex urban environments that are influenced by various socio-economic, ecological, and infrastructural factors. From designing effective city architecture, inspecting the effects of climate change, to managing transport and traffic systems are

among the numerous aspects in which these digital twins can assist with sustainable development in Zurich, Herrenberg, Helsinki, and Greece, according to the authors of [11][15][20][30]. Furthermore, the research papers propose the incorporation of citizen participation and engagement to gather diverse perspectives by providing interactive tools that allow residents to visualize and contribute their ideas to urban planning discussions.

2.3 Digital Twin in Traffic Management

The research done in the paper [59] discusses the limitations of existing digital twins in urban mobility, highlighting their limitations in scalability and simulation speed. Digital twins like Amodeus are often limited to smaller scales, lacking the capacity to simulate large-scale metropolitan systems. Traditional traffic simulators like VISSIM and SUMO focused on traffic flow and signal optimization but large-scale data integration. This paper introduced the Digital Twin for Urban Mobility Operating System (DTUMOS), an open-source framework that integrates an AI-based estimated time arrival model and routing algorithm, enabling it to handle large data volumes and simulate extensive urban areas. This approach supports advanced algorithms for fleet management, dynamic pricing, and mobility-as-a-service models.

Digital twin (DT) technology has become popular as a cutting-edge method to improve intelligent traffic management. With real-time modeling, predictive functionality, and system optimization, DTs provide a major edge over typical traffic simulations. Digital twins replicate transportation networks to predict possible problems and enhance traffic flow, making them useful for modern traffic problems. According to [18], it presents a three-layer design theory (DT) architecture that enables real-time data integration and analysis. It consists of three layers: data access, computational simulation, and application management. DT applications in transportation are supported by technologies like data processing, cloud computing, and data mining. However, persistent difficulties exist in handling large-scale traffic datasets, requiring further development. Despite these challenges, DTs offer promise for advancing intelligent expressways, autonomous driving, and ITS.

The study done in [45], explores the use of Digital Twin (DT) technology in improving adaptive traffic signal control (ATSC). DT creates a digital replica of a real-world traffic system, which will offer a more dynamic approach than traditional systems that rely on connected vehicle (CV) data. This allows for real-time data sharing, enabling more sophisticated responses to changing traffic conditions, especially in heavy congestion situations. The research presents a novel technique that uses DT's capabilities to combine waiting times from both upstream and near intersections, enhancing overall traffic performance and achieving a more accurate distribution of delays. The case study demonstrates the efficiency of DT-based ATSC compared to conventional techniques. The study suggests DT technology has great promise for managing traffic in cities, paving the way for more effective urban transportation systems.

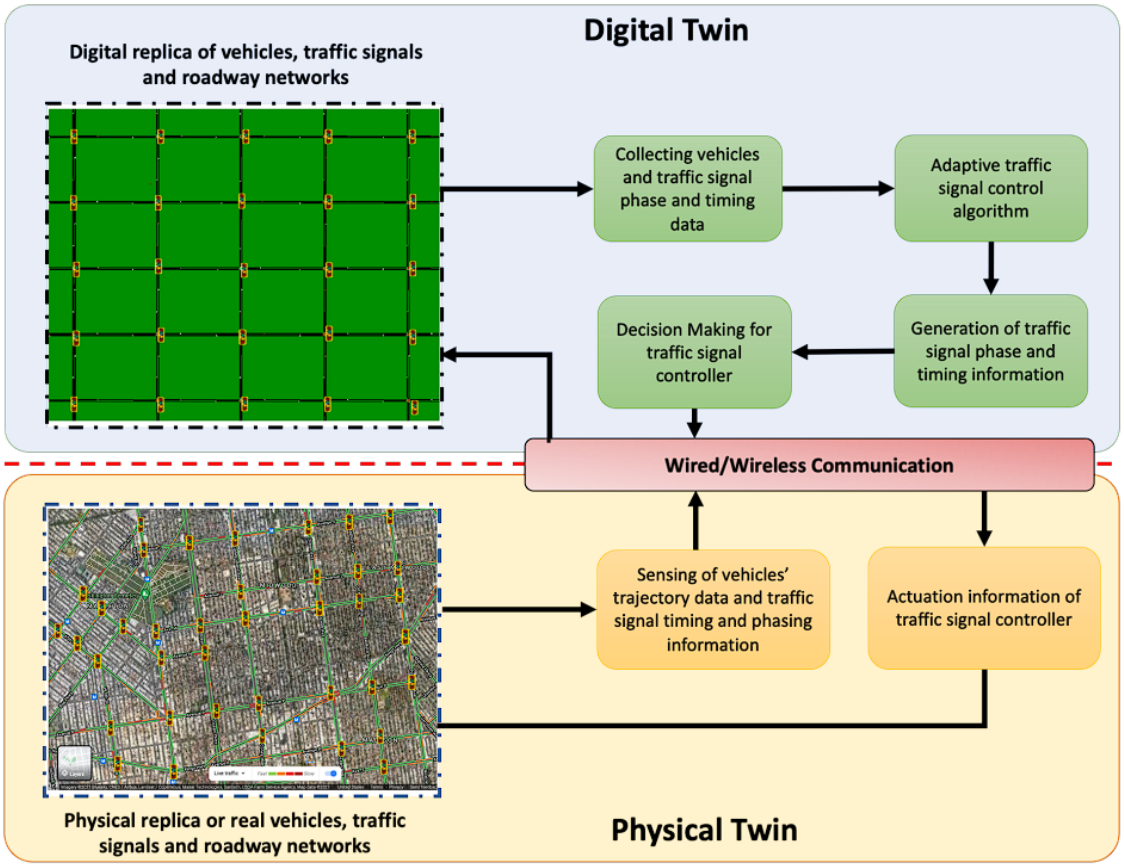

Figure 2.4: Twining of digital and physical space of adaptive traffic signal control systems [45].

The research presented in [54] highlights the growing integration of social, informational, and knowledge components in smart cities, particularly in transportation. It emphasizes the importance of Digital Twin (DT) technology in intelligent traffic management, highlighting its potential for enhancing sustainability, effectiveness, and optimization. Despite extensive research, most studies have focused on the technical advancements of DT frameworks rather than their direct application to traffic control. Integrating DT with the Internet of Things (IoT) has shown potential in addressing environmental concerns and transportation congestion in populated areas. Top nations are using IoT, Spatial Data Infrastructure (SDI), and Building Information Modeling (BIM) to create smarter urban environments. However, challenges such as data accuracy, synchronization between real and virtual models, and sophisticated modeling methods remain. The paper underscores the need to overcome these obstacles to fully realize DT technology's potential in urban traffic management.

This article [70] discusses the use of automated traffic signal performance measurement (ATSPM) using digital twin technology to improve traffic management. Digital twins are virtual versions of real-time traffic systems, providing improved tools for tracking and evaluating traffic signal efficiency. The study uses ATSPM-in-the-loop in a realistic simulation environment to assess and optimize traffic signal designs before deployment. The system offers comprehensive metrics beyond typical measures like average delays, enabling the forensic study of traffic light effectiveness

through accurate modeling of traffic events and connected car data. The proposed structure links traffic light operation and design, offering full spectrum solution assistance by integrating the real ATSPM system into the simulation. This system can provide a new traffic signal indicator and identify overloaded areas. According to the analysis in this article, digital twins like ATSPM-in-the-loop have a lot of promise to enhance intersection design and adaptive traffic management, which will make them an important tool for urban mobility in the future.

One of the main problems is traffic congestion at signalized crossings, which reduces the efficiency of urban mobility and causes delays. There are now three types of traffic signal control (TSC) systems: actuated, adaptive, and fixed-time. Seeking to improve intersection performance by dynamically modifying signal timings based on real-time traffic data, adaptive TSC systems differ from fixed-time systems in that they are not sensitive to real-time situations, while actuated systems depend on pre-optimized signal timing plans. This paper [63]claims that Digital Twin-based Adaptive Traffic Signal Control (ATSC) frameworks, like the suggested DT1 (Digital Twin 1) and DT2 (Digital Twin 2) algorithms, can improve user satisfaction by more evenly distributing delays among various approaches at intersections and significantly reduce delays when compared to conventional methods. While DT2 uses the delay of every car that happens in every approach connected to the target junction as well as the immediately adjacent intersection, DT1 uses the delay that every vehicle experienced from all approaches connected to the target intersection.

The study in [51] presents that optimizing traffic signal control can lower fuel consumption and delay when combined with Digital Twin (DT) technology and decentralized graph-based multi-agent reinforcement learning (DGMARL). By using DT simulations, DGMARL agents are able to observe traffic patterns and synchronize changes to the signals at intersections, resulting in enhanced adaptive traffic control efficiency. The approach in this paper [51] offers a promising alternative for intelligent transportation systems by addressing the flexibility and adaptability limit of centralized systems through the integration of real-time data and predictive models. A traffic simulation tool called PTV-Vissim was used to evaluate the approach.

This article [58] states that the Chattanooga Digital Twin (CTwin) platform optimizes traffic conditions in Chattanooga, Tennessee, by utilizing real-time data and interactive analytics to provide an integrated framework for urban mobility management. Cyber-physical traffic signal control, predictive analytics, and situational awareness are made possible by combining Internet of Things sensors with data from various urban areas. Compared with traditional digital twin models, CTwin is an inclusive model that takes into account various factors like traffic risks and the weather that have an impact on urban mobility. CTwin is a scalable architecture that is flexible and modular. Flexible designs are useful as CTwin that supports well-informed decision-making when deciding on the scope of implementation and for sustainable urban transportation systems. This is the procedure that follows.

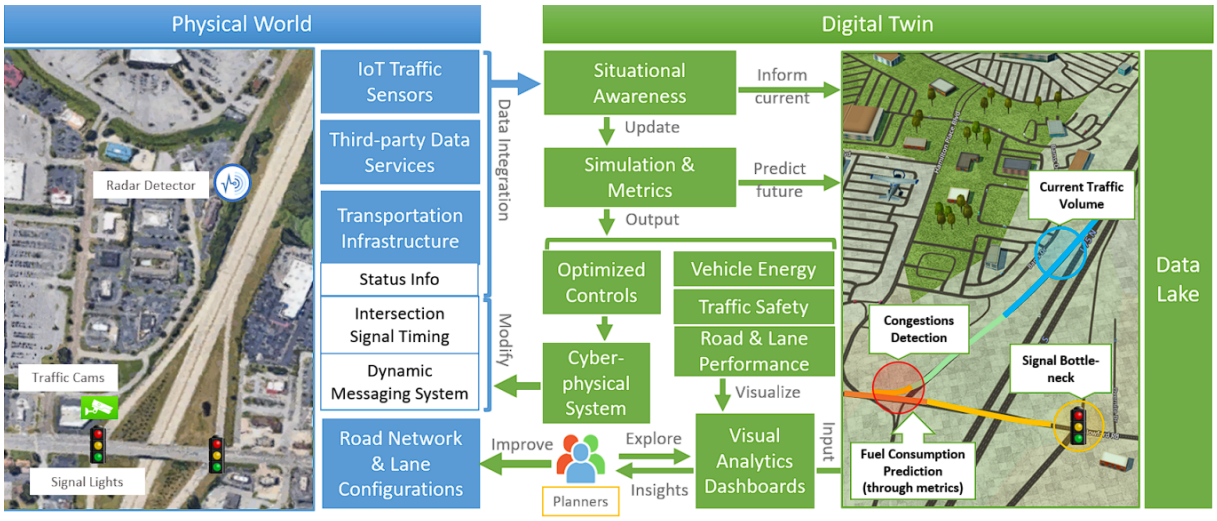

Figure 2.5: TA smart mobility management paradigm enabled through CTwins workflow: from virtual to reality [58]

To address the primary issues facing transportation networks and further their practical development, the article [24] discusses the use of digital twins and artificial intelligence in intelligent transportation systems. The researchers examined earlier data, including sensor data recorded in the information system and deployed in the road transport network, using artificial intelligence algorithms. Furthermore, by combining these services with digital twins of the road network and prediction analysis, it would be easier to plan for the growth of the transportation network and see the trouble spots in the city. The results of the study [24] fully accomplish the three primary roles of the intelligent transportation system, including managing emergency services in the event of an accident, arranging effective traffic flow, optimizing traffic dispersion, developing the transportation infrastructure, and informing interested parties about road conditions.

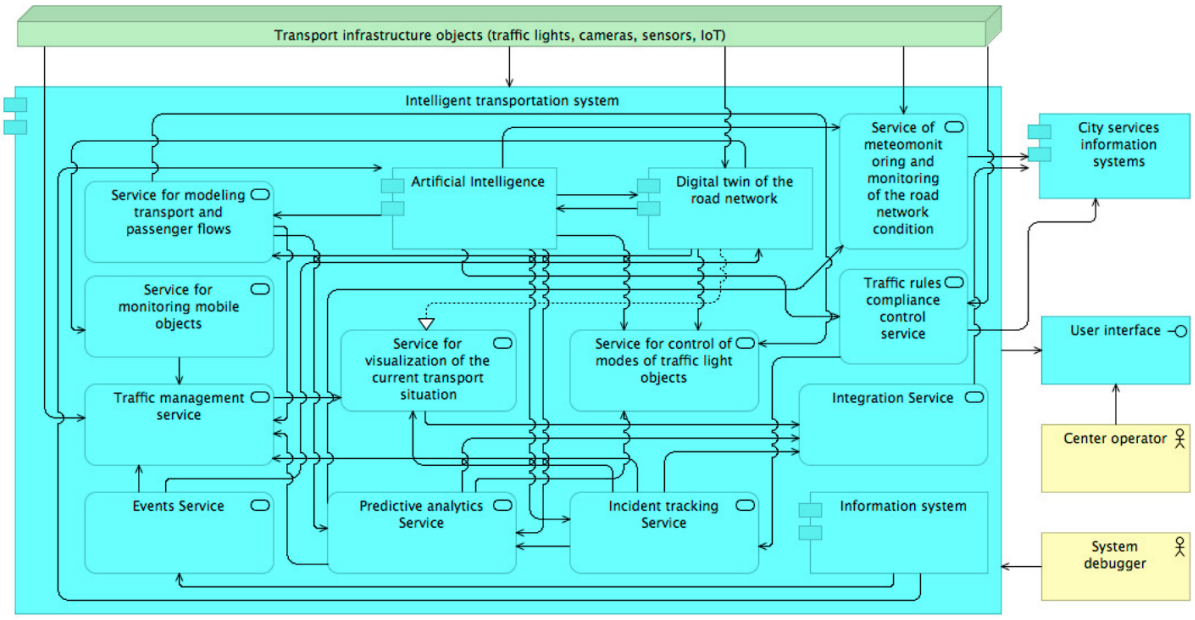

Figure 2.6: Reference Model of Intelligent Transport System [24].

The comparison of five significant studies on digital twins and their use in traffic management is provided below:

Paper	**Focus**	**Capacity**	**Real-Time Integration**	**Simulation Capabilities**	**Adaptability**
DTUMOS, digital twin for large-scale urban mobility operating system [59]	The focus is on routing algorithms, dynamic pricing, and AI-driven fleet management for broader urban mobility.	The system has limited capacity and is primarily used for simulating large urban areas, but it struggles with real-time synchronization	Mostly uses estimated data and AI modeling instead of complete real-time data.	Strong in predictive simulation for large-scale urban areas.	The fleet management and service models are quite flexible, but they are not as flexible when it comes to local behavior and intersection.
Three-Layer DT Architecture for Traffic Management [18]	Dynamic traffic control using computational simulation and real-time data access.	Suited for medium-sized systems, finds difficulties handling huge amounts of real-time data.	Integrates real-time data well through its layered architecture.	Focuses on current traffic problems and allows for the simulation of traffic flow in real-time.	Flexible when processing data in real-time, but not strong enough for major urban areas settings.

Paper	Focus	Capacity	Real-Time Integration	Simulation Capabilities	Adaptability
DT-based Adaptive Traffic Signal Control (ATSC) [45]	Improved traffic signal control by including real-time congestion response.	Scalable for intersection-based control systems but lacks capacity across large networks.	Strong integration of real-time data, especially at intersections.	Optimizing wait times at intersections with dynamic traffic signal simulation.	High adaptability at local intersection levels but struggles with broader coordination across multiple intersections.
Chattanooga Digital Twin (CTwin) [58]	Real-time data and predictive analytics are used to provide a comprehensive framework for urban mobility.	scalable architecture suitable for controlling many urban elements (such as traffic and weather).	Strong real-time data integration via IoT and sensor networks.	High-level simulation capabilities for various urban mobility factors.	Highly adaptable due to modular and flexible design.
ATSM-in-the-loop with Digital Twin [70]	Focus on evaluating and improving traffic signal performance with DT technology.	Limited to signalized area scales and intersections.	Significant use of real-time data, primarily for light control and traffic signals.	Simulations focused on optimizing traffic light performance and delay metrics.	Most adaptable for intersection management and traffic signal design.

Table 2.1: Comparative Analysis of Digital Twin Technologies for Traffic Management

.

Table 2.1 compares five papers based on key aspects such as adaptability, scalability, simulation capabilities, and real-time data integration. Although every research presents a different strategy for enhancing traffic management using digital twin technology, still scalability and handling large urban environments remain a consistent challenge across all studies.

2.4 YOLO in Vehicle Detection

In computer research, the most important phase is the identification of objects with reference to a vehicle and tracking since intelligent traffic management is being actively researched [17]. Vehicle detection is essential in traffic simulation systems because it provides precise information about vehicle types, positions, points of interest and movements. This information is necessary for simulating realistic traffic patterns, optimizing traffic control systems, and improving the efficiency of self-driving algorithms.Better urban mobility, congestion reduction, and traffic signal optimization all depend on real-time, dependable vehicle detection [22].Accurate vehicle detection is fundamental in traffic monitoring, autonomous driving, and intelligent transportation system development.

YOLO is ideal for vehicle detection in traffic simulations because of its real-time processing capabilities, high accuracy, and efficiency[3]. Its capacity to recognize various vehicle types (such as cars, buses, trucks, and motorcycles) in a single operation makes it excellent to evaluate complicated traffic conditions. The area of road traffic sight is currently the subject of much research. For examining the movement behavior of traffic objects, Murugan et al. [8] used the two-stage object detection algorithm R-CNN for vehicle identification in the traffic monitoring system. In order to improve performance for traffic sign identification by edge devices with self-driving cars, Liang et al. [35] created a sparse R-CNN that combines a coordinate attention mechanism with ResNeSt. This was done in response to the RCNN algorithm's low performance for small-target detection. The accuracy of object detection is somewhat improved using the R-CNN series method, but it detects objects more slowly than YOLO. The YOLO series method overcomes the target characteristics and speed problem and makes the generalized algorithm easier to understand by utilizing the concept of regression [32]. YOLO's durability in a variety of situations, such as changing lighting or weather, assures accurate detection in real-world scenarios. It specializes at managing many object classes at once, which is critical in jobs like traffic monitoring, where different objects such as automobiles, people on foot, and buses coexist [10]. Furthermore, its simplified architecture minimizes computing overhead, making it suitable for edge devices such as cameras or drones used in traffic monitoring systems [66]. These features make YOLO an effective tool for improving traffic simulation and management.

YOLO has been shown to be highly effective for vehicle recognition throughout its different versions, with each version improving to meet the needs of real-time and precise object detection in difficult circumstances [61][81]. From the initial iteration of YOLOv1, which introduced the grid-based detection approach [3], to the most recent iteration, YOLOv11, the algorithm has continuously improved its capacity to identify automobiles, trucks, and buses in a variety of scenarios, including obstruc-

tion, low light levels, and fast motion [61]. Advancements in YOLOv2 and YOLOv3, such as batch normalizing, anchor boxes, and multi-scale detection, significantly increased accuracy, allowing for better recognition of vehicles of varying distances and sizes [2][7]. YOLOv4 and YOLOv5 included new features such as Cross-Stage Partial Networks (CSPNet) and Path Aggregation Networks (PANet) [10][71], which improved the speed and accuracy of vehicle recognition in real-time applications such as traffic monitoring and automated driving.The change to YOLOv6 and YOLOv7 was designed at increasing computational efficiency, making these models acceptable for deployment in resource-constrained contexts without losing performance [34][41]. YOLOv8, YOLOv9, and YOLOv10 have continued this trend by improving scalability, feature extraction, and multi-task capabilities such as instance segmentation, all of which are critical for effective vehicle recognition in complicated scenarios [48][81][69]. Finally, YOLOv11 has cemented its status as the most advanced model for vehicle detection by introducing cutting-edge modules like Cross Stage Partial with Spatial Attention (C2PSA) and the efficient C3k2 block, which optimize spatial attention and improve the detection of small, overlapping vehicles without compromising processing speed [61].

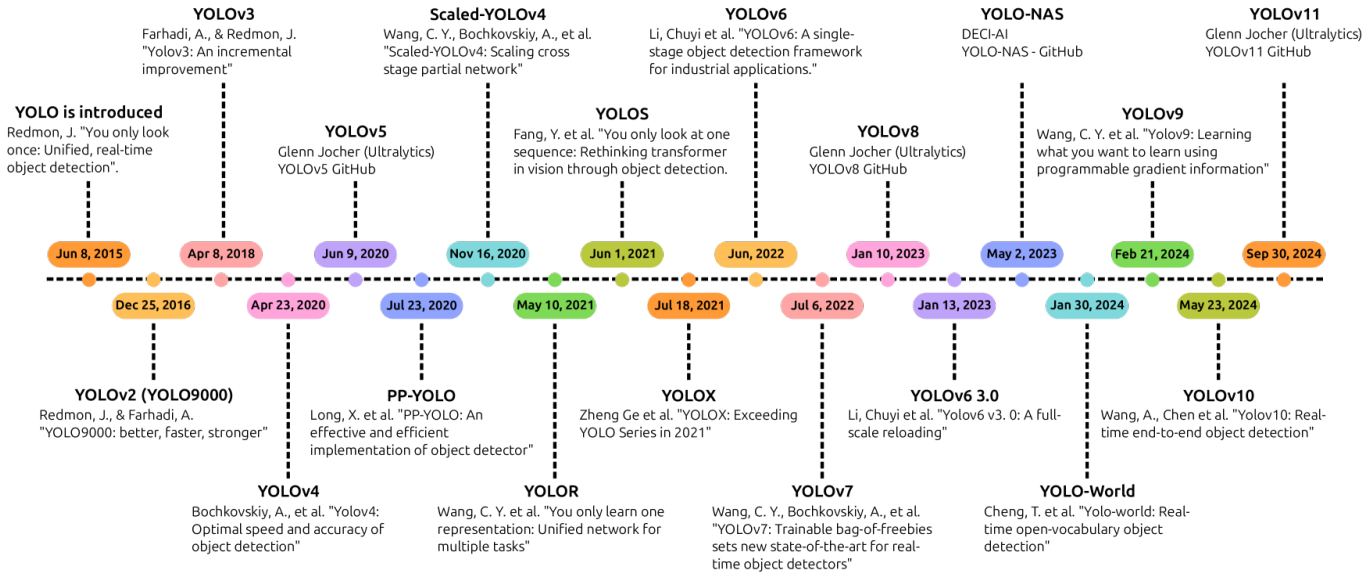

Figure 2.7: Evolution of YOLO Algorithms throughout the years. [69]

The Comparison of YOLO Versions and their Advancements for Object Detection in Vehicle Detection Tasks is Given Below:

YOLO Version	**Year**	**Key Features for Object Detection**	**Limitations**
YOLOv1	2016	Introduced grid-based detection, which allows for real-time detection.	Had difficulties with small or overlapping objects and lacked accuracy in localization.had difficulties with small or overlapping objects and lacked accuracy in localization.
YOLOv2	2017	High-resolution detection, batch normalization, and anchor boxes were added.	Limited in handling small objects and multi-scale detection.
YOLOv3	2018	Improved accuracy for small objects by introducing deeper feature extraction and multi-scale detection.	Increased computational demands.
YOLOv4	2020	Integrated CSPNet and PANet, balancing speed and accuracy.	Not ideal for deploying edge devices.
YOLOv5	2020	lightweight design and efficient pattern augmentation.	Less accurate for overlapping objects.
YOLOv6	2021	Better speed-accuracy trade-offs for real-time applications.	Improvements in marginal accuracy over YOLOv6.
YOLOv7	2021	Better speed-accuracy trade-offs for real-time applications.	Improvements in marginal accuracy over YOLOv6.
YOLOv8	2023	Added segmentation and tracking for video analysis.	High computational demands.

YOLO Version	Year	Key Features for Object Detection	Limitations
YOLOv9	2024	Enhanced tracking, small object detection.	Computationally intensive.
YOLOv10	2024	Optimized for edge devices, ultra-lightweight.	Lower accuracy for fine-grained details.
YOLOv11	2025	Adaptive attention and anchor-free detection for small/fast objects.	Represents the most advanced model to date with no significant limitations reported.

Table 2.2: Comparative Analysis of YOLO Versions for Object Detection

.

Because of its cutting-edge design and unmatched capabilities suited to contemporary object detection difficulties, YOLOv11 is the best option for vehicle detection. The Cross Stage Partial with Spatial Attention (C2PSA) module, which YOLOv11 introduces, builds on the advantages of its predecessors by greatly improving spatial attention in feature maps. Because of this, it is especially good at seeing tiny or overlapping cars in situations with heavy traffic. By adding the C3k2 block, efficiency is further maximized, enabling YOLOv11 to attain more accuracy while preserving real-time processing rates. It is adaptable to a variety of applications, from high-performance computers to lightweight edge devices, because of its capacity to scale across various computing contexts [61][69]. The effectiveness of YOLO11 in vehicle detection, emphasizing its capacity to manage intricate and instantaneous detection situations. YOLO11 aims to increase detection accuracy for a variety of vehicle types, including smaller and partially occluded objects, while preserving efficiency appropriate for real-time applications like autonomous driving and traffic management by utilizing deep learning breakthroughs and incorporating architectural innovations[61].

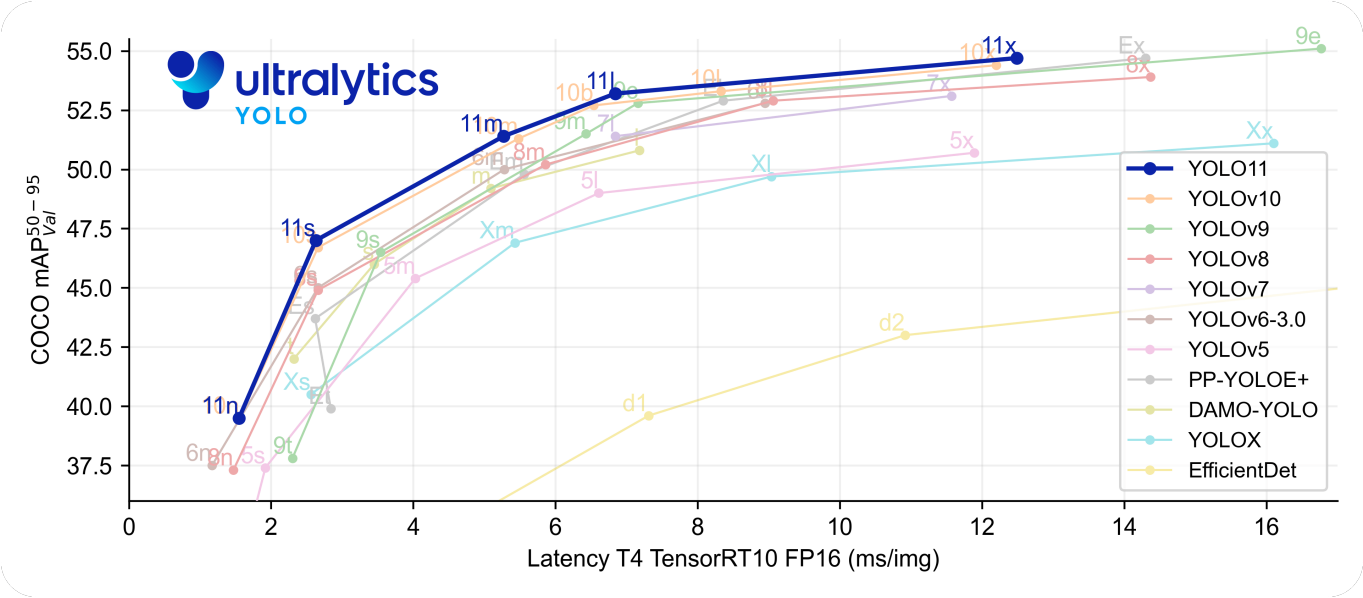

Figure 2.8: Performance of YOLO models [80]

2.5 Traffic Simulation

Using simulations and artificial infrastructure, traffic management has advanced in a number of ways. To optimize traffic routing, this paper [9], proposes an Intelligent Traffic Management System (ITMS) that combines the Dijkstra algorithm with a Deep-Neuro-Fuzzy model. These models are validated by simulation tools such as SUMO, which simulate real-world traffic conditions. A new Graphical User Interface (GUI) is also created to regulate the simulation input. For the optimal route, three different algorithms, including CHWrapper, A*, and Dijkstra, are used to trace. Moreover, traffic signal improvement, route planning, and vehicle communication are all enhanced by the integration of AI, IoT, and neural networks.

According to the paper [12], traffic simulation models are essential for optimizing urban transportation systems because they allow academics to analyze and solve traffic-related issues without having to implement real-world implementations. Microscopic, macroscopic, and mesoscopic models are the three categories into which these models fall. Mesoscopic models find a balance between the three, macroscopic models handle traffic flow on a broader scale, while microscopic models concentrate on interactions between individual vehicles. Various models have different benefits and drawbacks based on how complicated the transportation problems under research are.

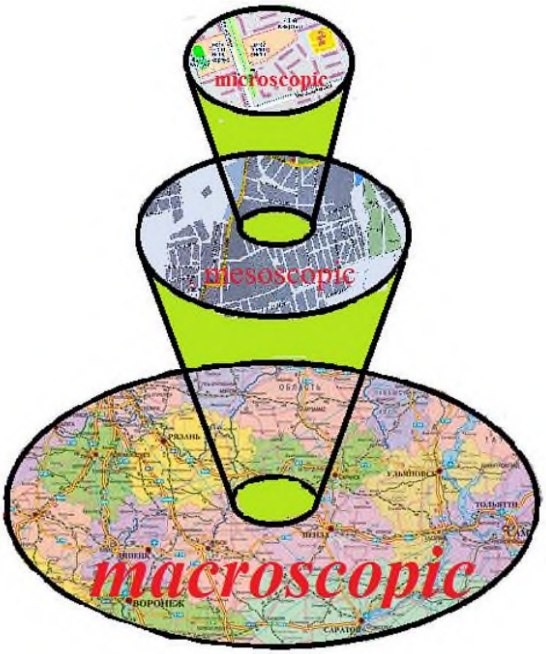

Figure 2.9: Classification of models of traffic flows [12]

Different traffic prediction and simulation models have been developed as a result of improvements in intelligent transportation systems (ITS). Reviewing non-parametric models, the paper [28] highlights their benefits, drawbacks, and uses in both short-term and long-term traffic forecasting. It draws attention to the possibility of complex models, which combine non-parametric methods for increased accuracy. Furthermore, the paper [28] examines three main traffic simulation algorithms: gap acceptance, lane changing, and car following. It also compares the features and flexibility of various tools, including SUMO, MATSim, AIMSUN, and VISSIM, and provides recommendations for choosing the right simulator based on particular needs.

Several attempts to combine SUMO with modern simulation systems, such as Unity, to enhance the reality of traffic simulations. Sumonity is introduced in paper [77], which aims to bridge the gap between SUMO and Unity by improving vehicle dynamics through the use of pure search control methods. By improving sideways driving behavior, this method overcomes earlier restrictions in vehicle control in co-simulation frameworks. Traffic signal synchronization and pedestrian modeling still provide difficulties, though. Unlike previous SUMO-Unity integrations, which sometimes lacked such expertise, Sumonity uses a structural vehicle control method.

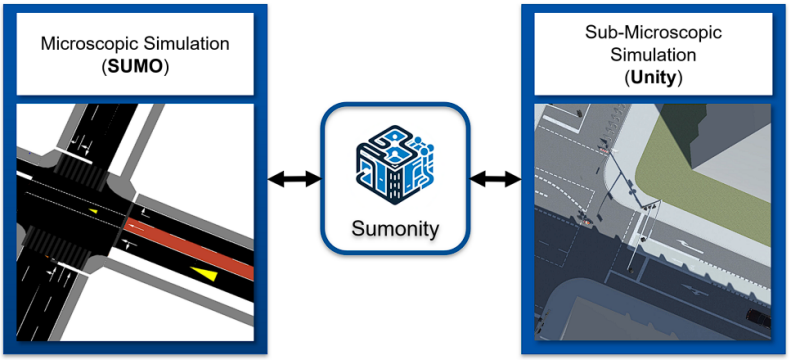

Figure 2.10: Illustration of the basic architecture of the Sumonity, SUMO/Unity interface. [77]

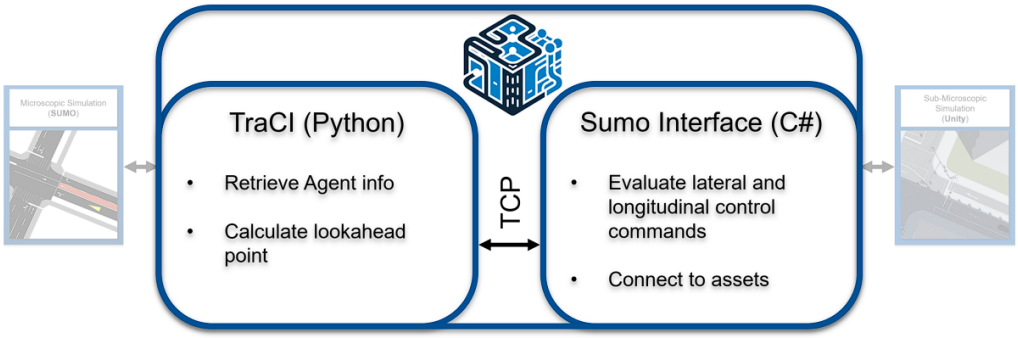

Figure 2.11: Illustration of the detailed architecture of the Sumonity, SUMO/Unity interface [77]

On the other hand, SUMO2Unity, an open source co-simulation tool designed to facilitate SUMO-Unity integration for traffic safety research, is presented in paper [74]. By creating a 2D-to-3D environment conversion, VR-based driver simulation, and wider research applicability, SUMO2Unity promotes comprehensive simulation in contrast to Sumonity, which focuses on sideways control enhancements. Although SUMO2Unity has limits in pedestrian modeling and weather condition integration, it offers a structured, expanded framework for a variety of research applications, while Sumonity aids in the improvement of vehicle behavior.

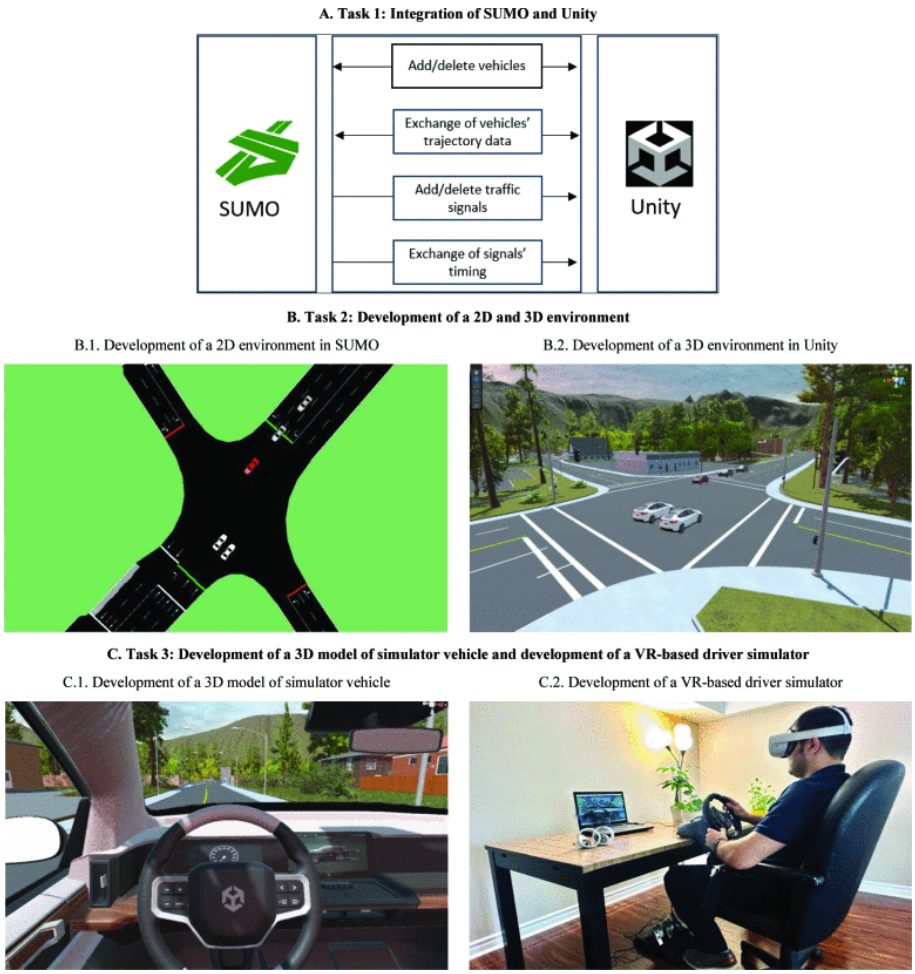

Figure 2.12: Tasks Required for Tool Development [74]

Both papers emphasize the need for SUMO-Unity co-simulation; however, paper [77] focuses on improving vehicle control, while paper [74] provides a more comprehensive open-source approach to facilitate further study.

2.6 Digital Twin in Extended Reality

The integration of Digital Twin Technology with extended reality (XR) can be revolutionizing, offering enhanced visualization and interaction. This combination offers improved analyzing capabilities that go beyond traditional digital twin applications providing opportunities for better urban design and management [40]. The paper proposes the exploitation of the human collaborative intelligence (HCLINT) framework for empowering Digital Twins, making them more immersive. Experimenting with a simple use case, the authors use their methodology to develop an interactive Digital Twin for a digitized collection of family photo albums implemented in AR and VR, analyzing the effectiveness of the immersive environment.

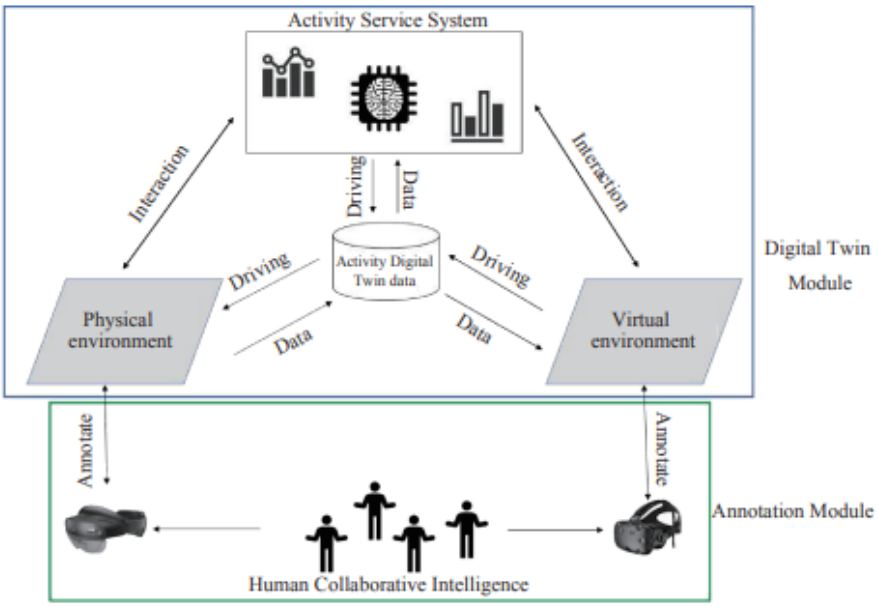

Figure 2.13: Main components of a HCLINT-DT framework [40]

The literature review compiled in [33] discusses how AR is used to enhance visualization of data associated with Digital Twins, allowing users to interact with and understand complex information more intuitively. AR applications provide guidance to users as they navigate tasks or processes related to Digital Twins, improving efficiency and accuracy in operations, in turn enabling users to manipulate and engage with virtual representations of physical assets in real-time with interactive experiences. Although there has been a lot of research, the authors of [33] underlined the significant gap in user-centered research, with very few studies involving users in the development and evaluation of AR applications for Digital Twins.

On the other hand, the research paper [57] highlighted experiments conducted by different authors, one of which used a DT of a powered wheelchair purposed to train untrained drivers. Comparing three groups of participants, a control group, a desktop group, and a VR group, it was found that the VR group had the highest evaluation scores. Additionally, the significant advantages of using virtual reality with digital twins is explored in the literature review synthesized in [36] including low-cost solutions, advanced immersive visualizations and direct, natural interaction. The integration of VR with digital twins allows for improved data collection and control mechanisms, facilitating better monitoring and management of systems. Furthermore, VR experiences can be designed to be flexible and portable, enabling users to access and interact with digital twins from various locations. This will allow teams to work together effectively and provide real-time feedback and interaction when physically apart.

Chapter 3

Background

3.1 Dataset

The Poribohon-BD dataset [16] was selected for our thesis due to its relatively larger size compared to other publicly available datasets from Bangladesh. It comprises 9,058 labeled and annotated images representing 15 distinct categories of native Bangladeshi vehicles.

3.1.1 Dataset Description

The data were collected primarily from highways in Bangladesh, with 1,791 images generated through augmentation, accounting for nearly one-fifth of the dataset, and 4,000 images sourced from Facebook (social media). The dataset is organized into 16 folders (classes), 15 of which represent unique vehicle categories, while the last contains mixed-class vehicles. Figure 3.1 presents the distribution of images across different vehicle categories in the Poribohon-BD dataset. The frequency of images for each vehicle class varies significantly, as some vehicle types are more commonly observed on the roads than others.

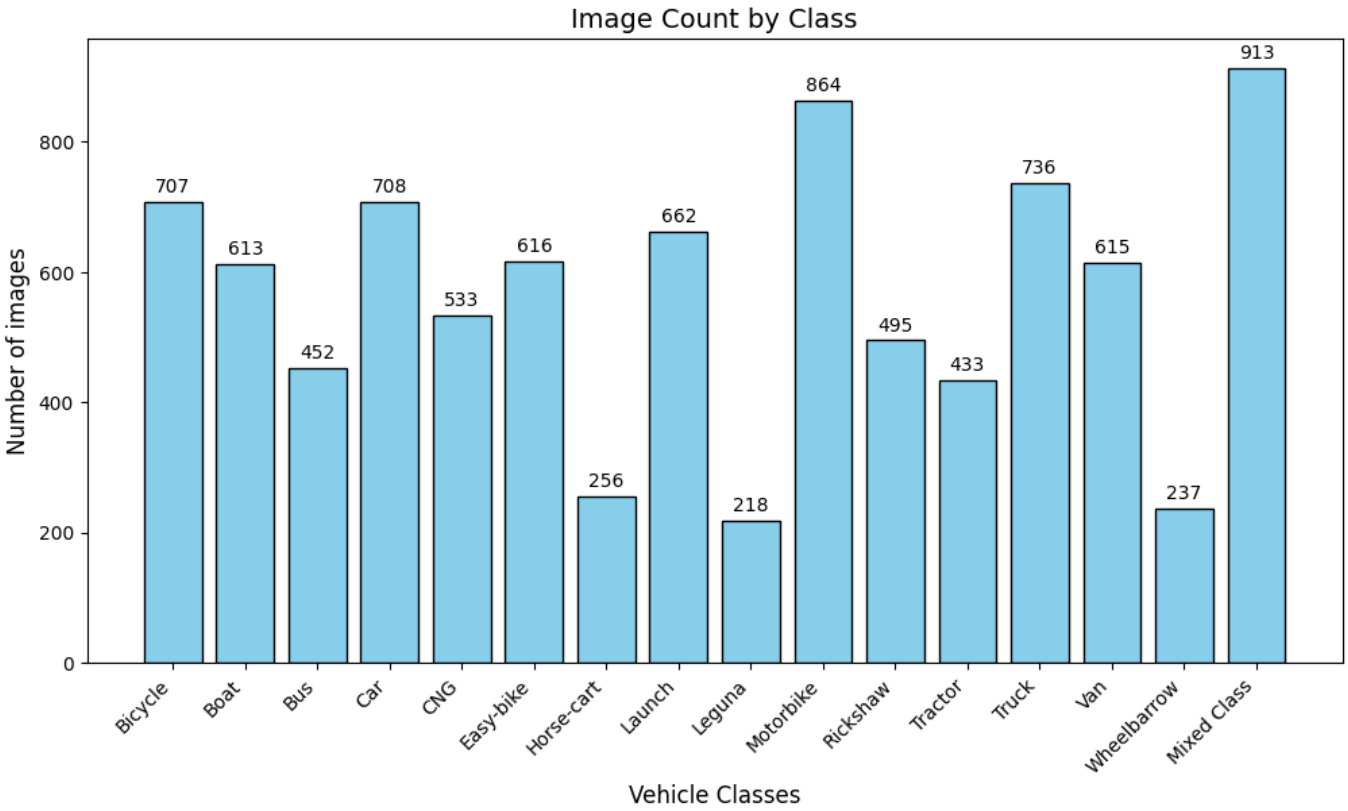

Figure 3.1: Class balance of Poribohon-BD dataset

However, the presence of a vehicle class within a specific folder does not necessarily imply that all annotations within that folder belong exclusively to that class. There are a total of 26,851 appearances that have been annotated in the dataset. Figure 3.2 presents the number of appearances each class has in the dataset.

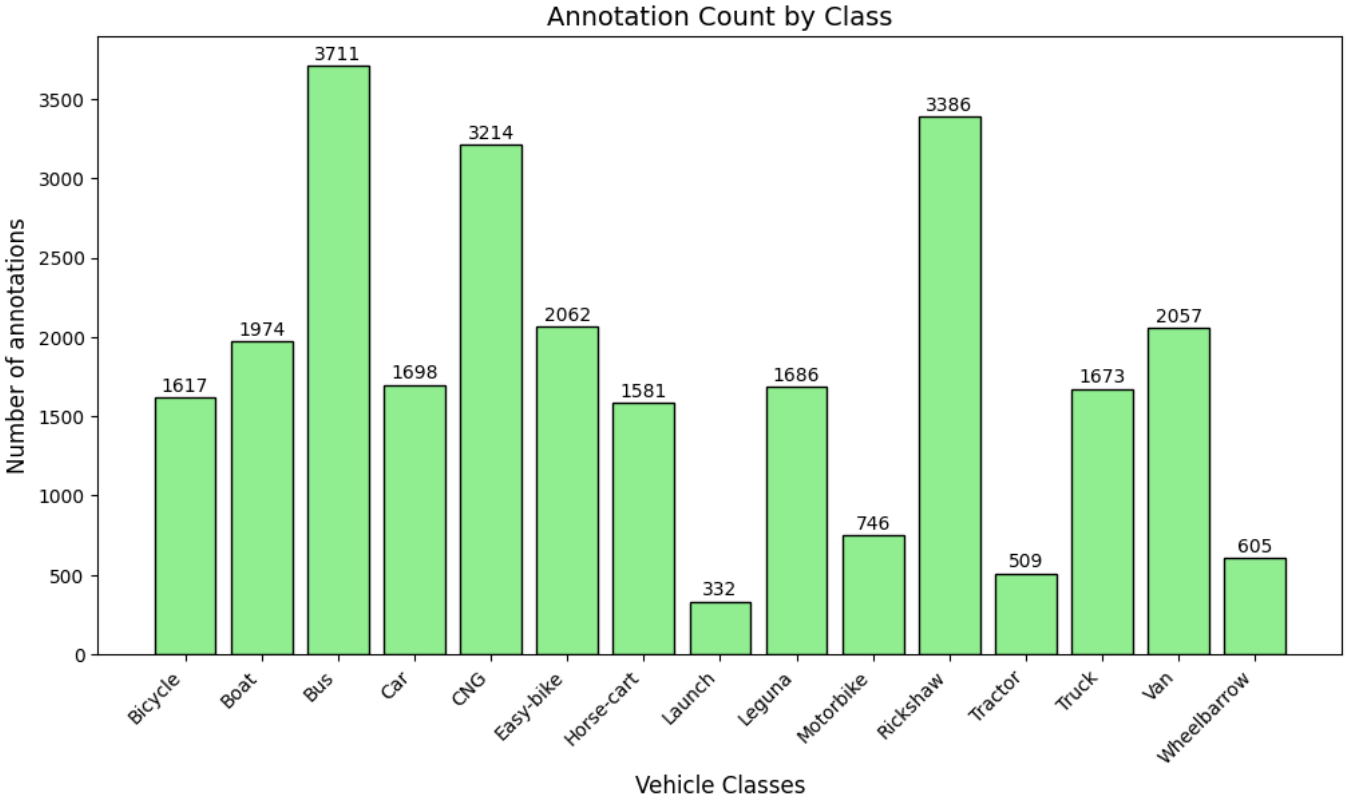

Figure 3.2: Annotation balance of Poribohon-BD dataset

To ensure privacy, human faces in the images were blurred. The dataset features a wide range of vehicle poses, views, and angles. Additionally, it includes images captured under varying lighting conditions—such as daylight (sunny), nighttime, and low-light environments—as well as some weather conditions, including rain. The data were collected primarily from highways in Bangladesh, with 1,791 images generated through augmentation, accounting for nearly one-fifth of the dataset, and 4,000 images sourced from Facebook (social media).

3.2 YOLO

Object detection is considered one of the most complex challenges in computer vision, as it involves both recognizing and localizing objects within the same image. In 2015, Joseph Redmon et al. introduced YOLO (You Only Look Once), a groundbreaking real-time object detection framework that significantly improved efficiency and accuracy in this field [3]. YOLO remains one of the most versatile and high-performing real-time object detection algorithms. Unlike conventional methods that apply models to images at various scales and positions, YOLO employs a single convolutional neural network (CNN) to process the entire image at once. This design enables rapid processing by segmenting the image into regions and simultaneously predicting bounding boxes along with their associated probabilities [52].

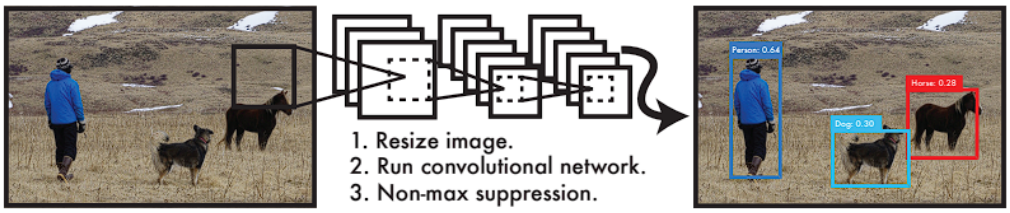

Figure 3.3: The YOLO Detection System.[3]

YOLO's architecture approaches object detection as a regression problem, streamlining and accelerating the detection pipeline. Its efficiency and straightforward design make it particularly well-suited for real-world applications, enhancing scalability and adaptability. By combining real-time processing, high accuracy, and robust performance, YOLO has set a new standard for object detection. Its continuous advancements and user-friendly structure have solidified its role in computer vision, making it an optimal choice for contemporary applications [76]. As a single-stage detector, YOLO integrates object localization and classification within a unified neural network, addressing key limitations of traditional multi-stage approaches [73]. This fully differentiable framework allows for end-to-end training, real-time detection, and reduced inference time, establishing it as a transformative solution in object recognition [72]. Given its widespread applications, the YOLO model is continuously updated, with YOLOv11 being the most recent iteration. For this reason, we have selected the YOLOv11 model for our thesis.

3.2.1 YOLOv11 Architecture

The YOLO architecture consists of three main components: the backbone, which extracts features from image data using convolutional neural networks; the neck, which uses specialized layers to aggregate and enhance feature representations across different scales; and the head, which generates object localization and classification outputs based on refined feature maps [72]. Building upon this design, YOLO11 improves detection performance through parameter improvements and architectural advances. Among the main changes in YOLO11 are:

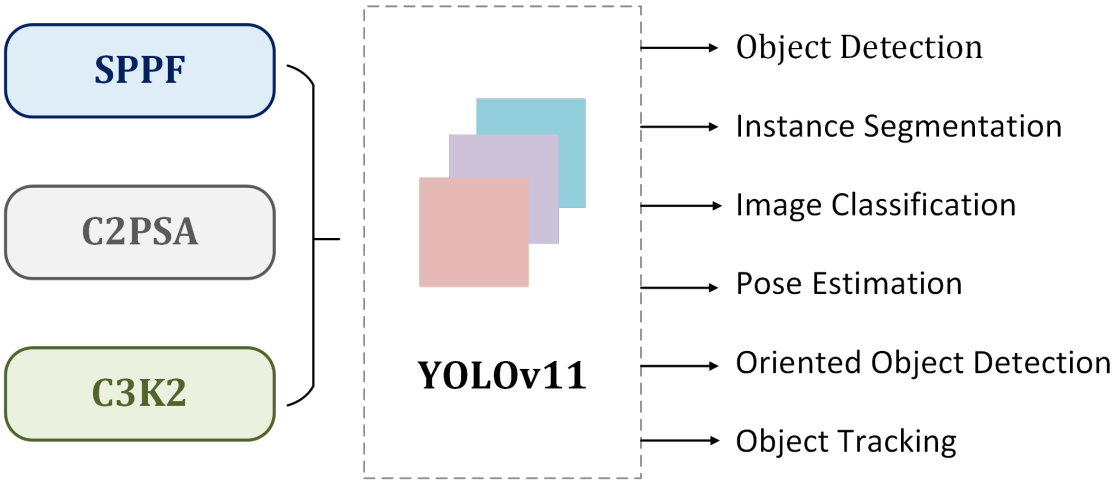

Figure 3.4: Key architectural modules in YOLOv11 [72]

YOLO11 is perfect for real-time tasks like vehicle detection because it integrates additional layers, blocks, and optimizations that improve detection accuracy and computing efficiency [61].

Backbone

The backbone of YOLO11 is responsible for extracting features from the input picture at different scales. This includes a sequence of convolutional layers and custom blocks that create feature maps of different resolutions [72][61]. Also YOLO11 introduces the C3k2 block while maintaining the Spatial Pyramid Pooling Fast (SPPF) block from earlier versions, as well as major enhancements to the C2PSA block [68].

Convolutional Layers

YOLOv11 uses initial convolutional layers to downsample the picture while keeping a structure similar to its predecessors. These layers, typically gradually decrease spatial dimensions while increasing the number of channels, serve as the basis for the feature extraction procedure [72][61]. This algorithm starts by downsampling the input picture using a sequence of convolutional layers:

$$\text{Conv}_1 = \text{Conv}(I, 64, 3, 2) \tag{1}$$

$$\text{Conv}_2 = \text{Conv}(\text{Conv}_1, 128, 3, 2) \tag{2}$$

C3k2 Block
YOLO11 features the more efficient C3k2 block, which is based on the Cross-Stage Partial (CSP) network, in place of the C2f block used in YOLOv8 [75]. Two smaller convolutions (kernel size = 2) make up the C3k2 block, which lowers computational costs without sacrificing performance [61][68]. This block is represented by the following equation:

$$C3k2(X) = \text{Conv}(\text{Split}(X)) + \text{Conv}(\text{Merge}(\text{Split}(X))) \tag{3}$$

In this case, Split(X) splits the feature map into two halves, one of which is processed by the bottleneck, and the results are then combined using Merge[61].

SPPF and C2PSA Blocks
In YOLO11, a new Cross Stage Partial with Spatial Attention (C2PSA) block is introduced following the Spatial Pyramid Pooling-Fast (SPPF) block, which is retained from earlier versions [75]. YOLO11 maintains the SPPF block, which performs efficient spatial pooling over multiple scales. It is defined as:

$$\text{SPPF}(X) = \text{Concat}(\text{MaxPool}(X, 5), \text{MaxPool}(X, 3), \text{MaxPool}(X, 1)) \tag{4}$$

The C2PSA block, which is also introduced in YOLOv11, improves spatial attention throughout the feature maps. This enhances performance on small and hidden objects by helping the model in concentrating on particular areas of the picture that are most important for the detection [61][72].

$$\text{C2PSA}(X) = \text{Attention}(\text{Concat}(X_{\text{path1}}, X_{\text{path2}})) \tag{5}$$

Neck

The neck of YOLO11 is designed to aggregate feature maps from various resolutions and send them to the detecting head. The C3k2 block is integrated into the neck by YOLO11 to improve feature aggregation performance and speed [61].

Feature Aggregation
The neck combines feature maps from several scales by applying concatenation layers and upsampling:

$$\text{Feature}_{\text{upsample}} = \text{Upsample}(\text{Feature}_{\text{previous}}) \tag{6}$$

$$\text{Feature}_{\text{concat}} = \text{Concat}(\text{Feature}_{\text{upsample}}, \text{Feature}_{\text{lower}}) \tag{7}$$

The use of the C3k2 block after concatenation ensures efficient feature aggregation.

$$\text{C3k2}_{\text{neck}} = \text{Conv}_{\text{small}}(\text{Concat}(\text{Feature}_{\text{concat}})) \tag{8}$$

Spatial Attention
By using the C2PSA block, YOLO11's neck also integrates spatial attention, which enhances the model's capacity to concentrate on the most crucial areas of the picture. This is especially helpful in scenarios that are congested and include overlapping items [61].

Head

The YOLO11 head is responsible for producing the model's final predictions. The head produces bounding boxes, class probabilities, and confidence scores much like earlier versions [61][72].

Detection Layers
YOLO11 has three scales of detecting layers to identify objects of different sizes: small (P3), medium (P4), and large (P5). Different feature map levels are processed by each scale, guaranteeing that the model can recognize both big and small vehicles [61].

$$\text{Detect}(P3, P4, P5) = \text{BoundingBoxes} + \text{ClassLabels} \tag{9}$$

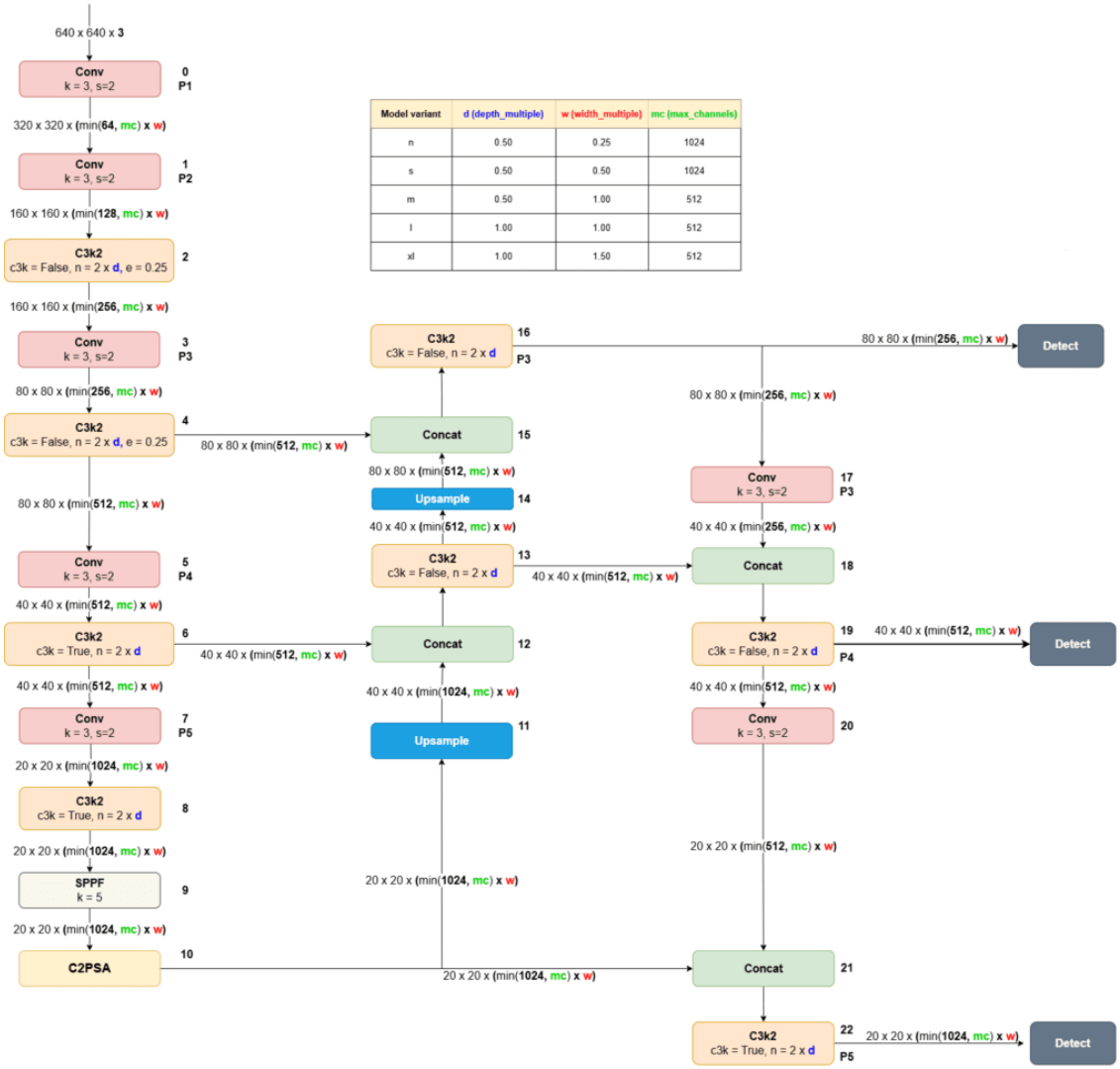

Figure 3.5: YOLO11 Architecture [82]

YOLO's development from the beginning to YOLOv11 shows a dedication to enhancing the effectiveness and precision of object detection. YOLOv11 marks a major advancement in real-time object identification, especially for intricate applications like autonomous navigation and vehicle detection, by combining cutting-edge elements like C3k2 and C2PSA blocks and refining current modules.

3.2.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is a widely used optimization algorithm for training machine learning models, especially deep neural networks. Unlike conventional Gradient Descent, which computes gradients using the entire dataset, SGD updates model parameters using a small batch or a single data point at a time. This approach enhances computational efficiency, making it particularly suitable for large-scale datasets [50].

SGD accelerates model convergence and mitigates overfitting by introducing randomness in weight updates. However, its stochastic nature can lead to fluctuations

during optimization. To counteract this, techniques such as momentum are often incorporated to stabilize learning.

3.2.3 Evaluation Metrics

Mean average precision (mAP) is commonly used to evaluate the performance of a YOLO model.

True Positive (TP): A correct detection of a ground-truth bounding box.

False Positive (FP): An incorrect detection where a non-existent object is detected or a bounding box is misplaced for an existing object.

False Negative (FN):A ground-truth bounding box that is not detected.

In object or vehicle detection, there is no True Negative (TN) because an infinite number of bounding boxes could potentially satisfy that condition. The classification of detection as correct or incorrect is determined based on Intersection over Union (IoU).

Intersection Over Union (IoU): IoU is a metric used to evaluate the overlap between the predicted bounding box and the ground-truth bounding box. By setting a threshold for IoU, correct and incorrect detections are determined. For example, an IoU threshold of 0.5 was applied in this experiment. If the predicted bounding box overlaps at least 50% with the ground-truth bounding box, it is considered a correct detection and vice versa.

Precision (P) and Recall (R): Precision refers to the ability of the model to correctly detect relevant objects, indicating the percentage of correct positive detections. In contrast, Recall represents the model's ability to detect all relevant instances, i.e., the percentage of correct positive detections out of all ground truth objects. The formulas for calculating precision and recall are:

$$P = \frac{TP}{TP + FP} = \frac{TP}{\text{All detections}} \tag{10}$$

$$R = \frac{TP}{TP + FN} = \frac{TP}{\text{All ground truths}} \tag{11}$$

Mean Average Precision (mAP): The formula for calculating mAP is,

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{12}$$

where N is the number of classes, and APi is the average precision of class i. We chose our model at the checkpoint where the model had the most mAP at IoU of

0.5, denoted as mAP50 or mAP0.5 throughout this report. Average Precision (AP) is the numerical value, the average of precision values across recall values between 0 and 1.

3.3 SUMO

Simulation of Urban Mobility (SUMO) is an open-source, high-performance traffic simulation software widely used for modelling and analyzing urban traffic systems. It is well known for microscopic and macroscopic simulation which provides accuracy in modelling real-world vehicles and traffic flow, while also offering a user-friendly graphical interface that allows even non-programmers to set up networks easily [64][44]. Developed by the German Aerospace Center (DLR), SUMO provides a versatile and scalable environment for researchers, city planners, and engineers to test traffic control strategies and optimize mobility solutions [21].

3.3.1 Key Features of SUMO

From basic traffic analysis and management to advanced AI-driven optimization, SUMO is a highly modular traffic simulation tool that supports a wide range of applications [25]. For digital twin based traffic management, SUMO is a powerful tool with multiple key features that include:

- **Microscopic, Mesoscopic, and Macroscopic Traffic Simulation**
  SUMO provides microscopic traffic simulation, where each vehicle is individually modeled with attributes such as acceleration, deceleration, lane-changing behavior, and route selection. Additionally, mesoscopic and macroscopic simulation modes allow large-scale modeling of traffic demand, network-wide vehicle flow, and congestion analysis, as done in [64][44].

- **OpenStreetMap (OSM) Integration to Create Real-World Road Network**
  SUMO has the ability to import real-world road networks from OpenStreetMap (OSM), GIS datasets, and other sources that can allow researchers to simulate actual city traffic conditions needed for a digital twin setup. The network creation process in SUMO typically involves extracting road networks from OSM using a tool called netconvert to convert raw OSM map data into SUMO-compatible road networks as shown in [21][64]. This can automatically extract road geometries (highways, streets, lanes, bike lanes), traffic signals and stop signs, intersection data, road hierarchy (e.g., primary roads, secondary roads, local streets), and so on [46]. Figure 3.6 shows a resulting route network in SUMO compared to its original map[21][1]. The graphical user interface SUMO provides, which allows easy monitoring of traffic flow simulation built on the network file, is shown in Figure 3.7 [46].

Figure 3.6: Original Map and abstracted road network of Jianghan Zone. [21]

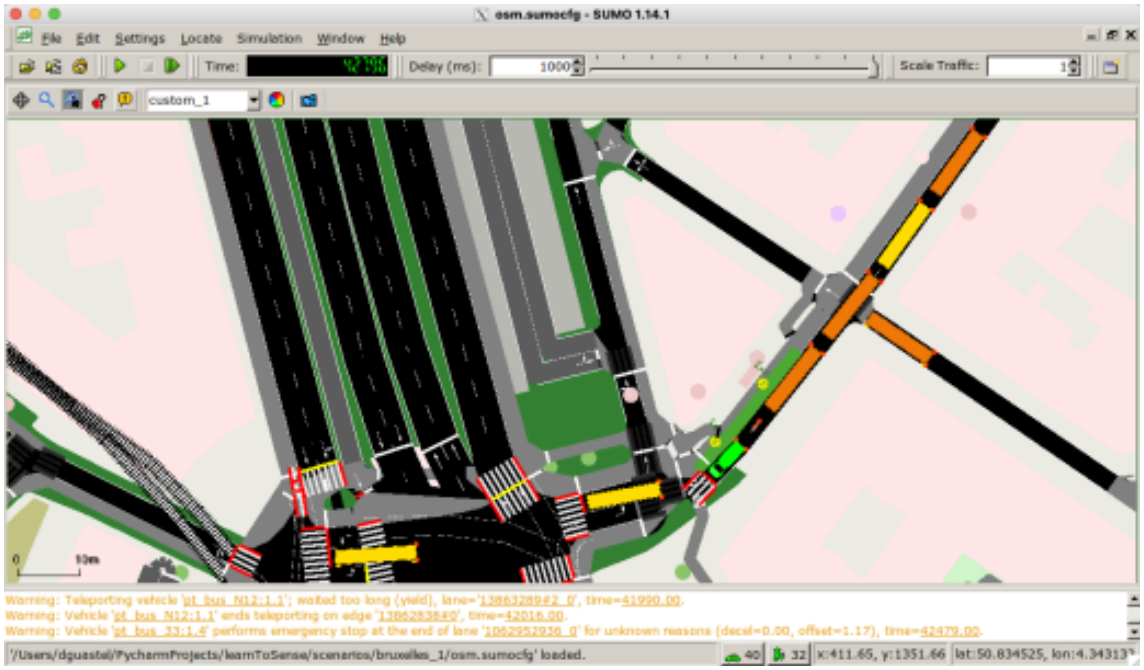

Figure 3.7: SUMO graphical user interface. [64]

Once the base road network is generated, additional real-world data, such as vehicles from traffic sensors, can be integrated into SUMO, mimicking real-world conditions [21][44][25]. SUMO also allows for manual editing using tools netedit (SUMO's GUI-based network editor), traffic signal adjustments, calibration of speed limits, lane connections, and road types to increase the accuracy of the simulation.

- **Real-Time Data Integration and Traffic Calibration**
  To enhance simulation accuracy, SUMO supports the integration of real-world traffic data from sources such as IoT sensors and GPS-based vehicle tracking, CCTV-based vehicle detection, and vehicle trajectory datasets [46]. Using the data, trip files can be generated where each vehicle trip is defined, which can then be used to generate route files containing all the valid routes using DUAROUTER, another attachment program in SUMO. By integrating SUMO with data from AI-based object detection models (e.g., YOLOv11), IoT sensors, and GPS tracking, digital twin simulations can:

  – Detect real-time vehicle density and predict congestion hotspots.

  – Optimize traffic signal control to improve flow efficiency.

  – Dynamically reroute traffic in response to road incidents or congestion.

  – Evaluate the impact of smart transportation policies before deployment

  In addition, SUMO allows real-time interaction with the simulation using tools such as TraCI (Traffic Control Interface), which is a Python-based API that provides access to a running road traffic simulation. This allows adjusting vehicle speed, route, lane-changing probabilities, and driver behavior parameters, dynamically, to closely match real-world traffic conditions [21][64][44].

  Figure 3.8 shows a flowchart process for building a complete simulation in SUMO.

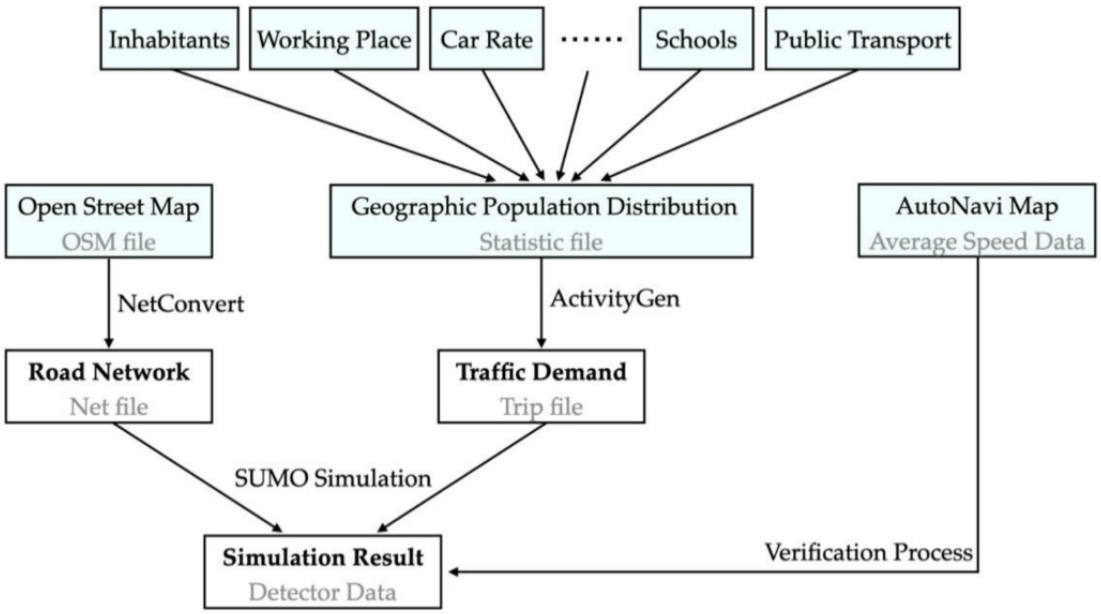

Figure 3.8: flowchart process for building a complete simulation in SUMO [21]

3.4 Unity

Unity is a powerful and versatile game engine used for developing interactive 2D and 3D applications, including video games, simulations, virtual environments, and graphical visualizations. It is considered beginner-friendly, efficient, and highly robust, thanks to its comprehensive suite of tools, plugins, and community support. The engine is known for delivering high-quality, stable results across multiple platforms.

With its robust physics engine, extensive asset store, and support for scripting in C#, Unity has become a preferred choice for developers across various industries, including gaming, education, architecture, and artificial intelligence. Our research utilizes Unity to create a traffic simulation that closely resembles the traffic system in Bangladesh. The complexity of traffic patterns in urban environments, particularly in a densely populated country like Bangladesh, requires an advanced simulation framework capable of managing dynamic interactions among various traffic entities. Unity's physics engine and scripting system make it an ideal platform for modeling vehicle movements, traffic congestion, and adherence to traffic regulations.

The expandable toolset makes this software ideal for simulations, as demonstrated by the researchers in [6]. The researchers discuss a port of SUMO to Unity and examine driving patterns and driver behaviors in realistic scenarios through various interactive components in the game engine. It primarily focuses on recording data for autonomous vehicles and the interaction of various moving parts in driving simulators.

Chapter 4

Research Methodology

4.1 Workflow

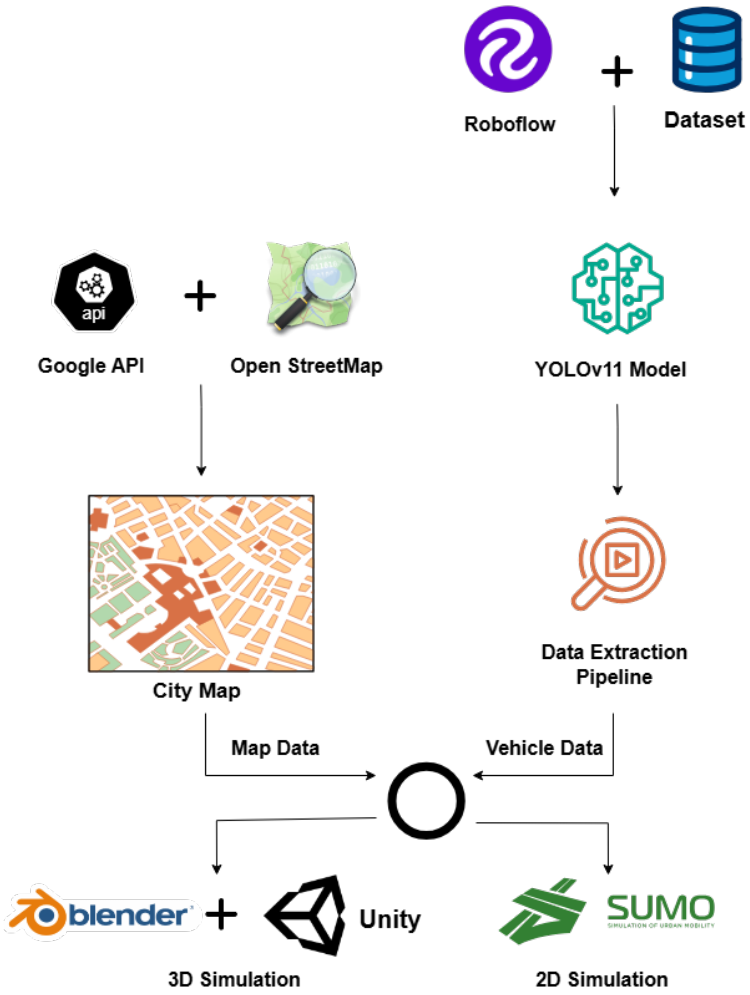

Figure 4.1: Work Plan Analysis

The workflow in Figure 4.1 represents an integrated approach for object detection, map data extraction, and simulation to analyze urban mobility.

- **Dataset Preparation and Object Detection:**
  The dataset is processed using Roboflow, which helps in augmenting and managing data for deep learning applications. This processed dataset is then used to train a YOLOv11 model, a real-time object identification system to precisely identify vehicle data and video streams.
- **Map Data Collection:**
  Geographical data is provided via the Google API and OpenStreetMap to create a city map that properly represents urban settings.
- **Data Extraction Pipeline:**
  The trained YOLOv11 model extracts vehicle-related information, including traffic density and movement patterns. This data is then processed through a Data Extraction Pipeline, which structures and refines it into vehicle data, making it suitable for further analysis and simulation.
- **Integration of Data for Simulation:**
  The extracted map data and vehicle data are integrated into a simulation environment to analyze urban mobility. Blender and Unity facilitate 3D simulations, creating realistic visual representations of the city and traffic movement, while SUMO (Simulation of Urban Mobility) is used for 2D simulations, focusing on traffic flow and vehicle behavior analysis.

4.2 Dataset Preparation

The dataset was prepared with the help of an end-to-end computer vision platform called Roboflow [65]. Which is widely used by many computer vision researchers and engineers.

4.2.1 Dataset Preprocessing

A version of the dataset was generated by applying auto-orient and resizing the images to 640x640. Additionally, images with no labels were filtered and removed. The dataset was divided into training, validation, and test sets.

Test-Validation-Train Split:
To assess the model's generalization capability, the dataset was divided into training validation and test sets. The training set enables the model to learn object features, while the validation set provides an unbiased evaluation of the model's performance. Lastly, the test set is used for the final evaluation of the model's performance. The dataset was divided following an 80-10-10% ratio.

Auto-orient:
Auto-orientation in image preprocessing involves automatically adjusting an image's

orientation using its metadata to eliminate EXIF rotations and ensure a consistent pixel arrangement.

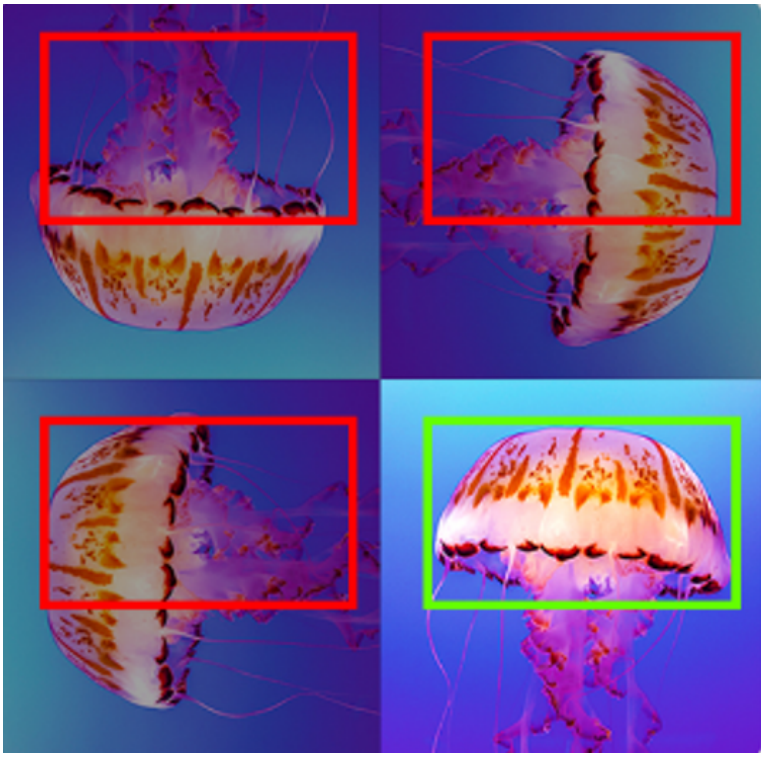

Figure 4.2: Auto orientation [65]

Resizing Image:
Image resizing is a fundamental step in image preprocessing that entails modifying an image's dimensions. In this study, the images were resized to 640 pixels in both height and width to enhance training speed and optimize storage efficiency

Figure 4.3: Image resized to 640x640 [65]

Filter-Null:
To preserve dataset integrity, images without bounding box annotations were excluded. Unlabeled images can introduce noise, potentially causing inaccuracies during model training. Removing these instances ensures that the dataset consists only of high-quality labeled images, leading to a more reliable training process.

4.2.2 Dataset Augmentation

Three images per training example were created using rotation, shear, mosaic, and adjusting bounding box brightness. These augmentation techniques were applied solely to the training set, while the validation and test sets remained unaltered to maintain an accurate evaluation process.

Rotation:
Rotation involves turning an image around its center by a specific angle. It is applied in the dataset to simulate different perspectives and orientations. For our model to better handle camera roll, we applied a 15% right and left rotation on the images.

Figure 4.4: Image Rotation [65]

Shear:
Shear is a geometric transformation that skews or tilts an image along a specific axis. We applied a 10% vertical and horizontal shear to simulate perspective distortions, helping the model become more resilient to variations in subject pitch and yaw.

Figure 4.5: Horizontal Shear (top), Vertical Shear (bottom) [65]

Mosaic:
The mosaic technique merges several images into one composite image. In this case, we combined four images to generate new ones. Mosaic augmentation was applied to the entire test set to improve the model's performance in detecting smaller objects.

Figure 4.6: Mosaic augmentation [65]

Bounding box brightness:
Bounding box brightness involves adjusting the lighting or exposure within a specific region of interest marked by a bounding box. We applied a 20% variation in brightness and darkness to enhance the model's resilience to changes in lighting and camera settings.

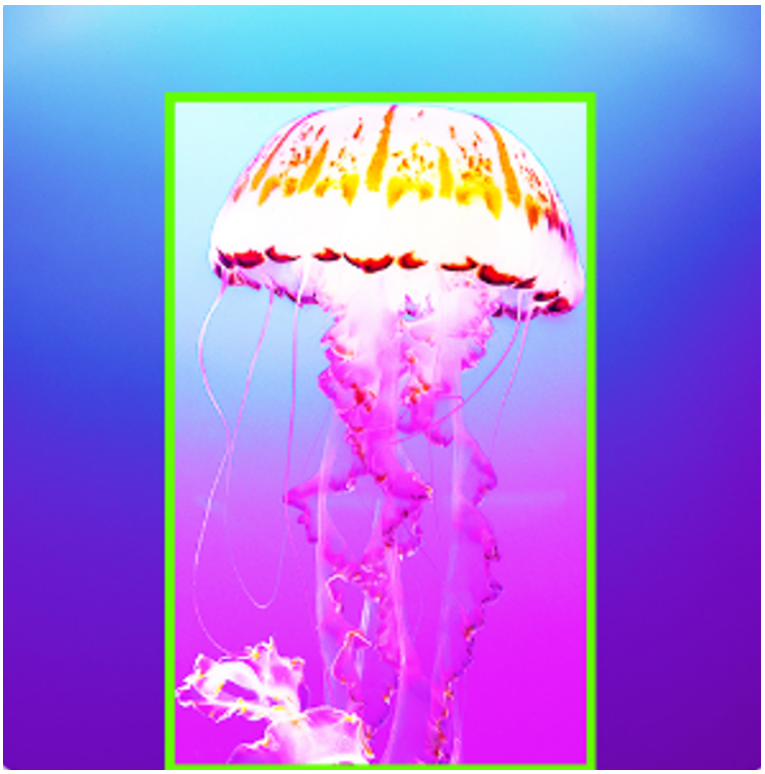

Figure 4.7: Bounding box augmentation [65]

4.2.3 Dataset Hosting and Exporting

The finalized dataset was uploaded to Roboflow for efficient management and accessibility. Labels were exported in the YOLOv11 format to ensure compatibility with our model.

Chapter 5

Experimental Evaluation

5.1 Experimental Setup

5.1.1 YOLO Model Experimental Setup

We applied transfer learning by initializing model weights with a pre-trained YOLOv11 model. This approach allowed the model to utilize features learned from larger datasets, improving detection accuracy on our dataset.

YOLOv11 Model Variant Selection

The YOLOv11m (Medium) variant was chosen for model training due to its balanced performance in both accuracy and computational efficiency. Among the various YOLOv11 models—Nano (N), Small (S), Medium (M), Large (L), and Extra Large (X)—the M variant offers an ideal compromise between lightweight deployment and high detection performance, making it well-suited for real-world applications where resource constraints are a factor. The YOLOv11m model ensures optimal performance without excessive computational demands, making it an excellent choice for vehicle detection tasks in the Poribohon-BD dataset and the best option for this experimental setup.

Training configuration

We trained the model using the Stochastic Gradient Descent (SGD) optimizer with a batch size of 32 for efficient GPU utilization. The image resolution was set to 640x640 pixels, and the initial learning rate (lr0) was 0.01, with a final learning rate (lrf) of 0.01.

Hyperparameter optimization

The hyperparameters were fine-tuned through a trial-and-error process based on validation performance. The model used a warmup momentum of 0.7 and a momentum of 0.85. Additionally, 10% mixup augmentation and 20% copy-paste augmentation were employed to improve the model's robustness against variations in lighting, perspective, and object occlusions.

Performance Enhancement

We enabled Automatic Mixed Precision (AMP) during training to reduce computational overhead while maintaining accuracy.

Model Visualization

During the training process, plots were generated to monitor key metrics such as loss, learning rate progression, and model performance across epochs. Additionally, precision-recall curves, precision, recall, F1 score, and confusion matrices were plotted to provide a comprehensive view of the model's performance.

5.1.2 Data Extraction Pipeline

In order to extract traffic data from captured footage, we need to pass it through detection and tracking using our own trained model on a custom dataset. The criteria for the captured footage are:

- The camera must be static.
- There should be a clear vision of both upstream and downstream of vehicles.

Upon fulfilling only these two requirements, we can obtain a highly accurate dataset on the movement of vehicles on the road. We are focusing on:

- **Velocity** (approximate)
- **Direction** (in/out)
- **Passing lane**
- **Vehicle Type**

We have prepared a simple GUI for the extraction process so that it can be used even without any programming knowledge.

Project Description

In order to set up the GUI and the detection pipeline, we have used the following libraries:

- **Ultralytics** (v8.3.69): Utilizes YOLOv11 model for detection and tracking.
- **OpenCV** (v4.11.0.86): Captures frames, adds annotations, and markers.
- **PyQT5** (v5.15.11): For setting up a comprehensive GUI with buttons, checkboxes, video players, etc.

The libraries used were in their latest version at the time of development (January 2025).
The tasks have been divided into several classes for ease of manipulation and management. The workspace mainly consists of the following classes/files:

- **VideoProcessingApp (Main)**: Handles the overall video processing workflow, managing interactions between different components like video playback, lane adjustment, and vehicle prediction.
- **VideoPlayer**: Responsible for playing the video stream and possibly displaying it for processing or user interaction.
- **LaneAdjustment**: Manages the lanes in the video, including enabling/disabling lanes, adjusting their positions, and ensuring they are correctly drawn or interacted with based on user input.
- **Predictor**: Handles vehicle prediction logic, likely using a model to track and analyze vehicles in the video based on frames.
- **Lane (Data Class)**: Holds data specific to each lane, such as start position, width, and enabled state. Also contains functions for drawing the lines on a frame, finding the start and end coordinates, etc.
- **Vehicle (Data Class)**: Holds data about each vehicle, such as ID, velocity, direction, lane, and label.
- **Util File**: Contains utility functions for tasks like annotation (adding labels/markers on frames), tracking vehicles across frames, and counting vehicles. Some of the functions used for velocity determination, drawing bounding boxes, and annotations are inspired by MuhammadMoinFaisal (GitHub) [67].

Video Selection

By clicking on the "Browse Video" button in the GUI, we can easily select our recorded video.

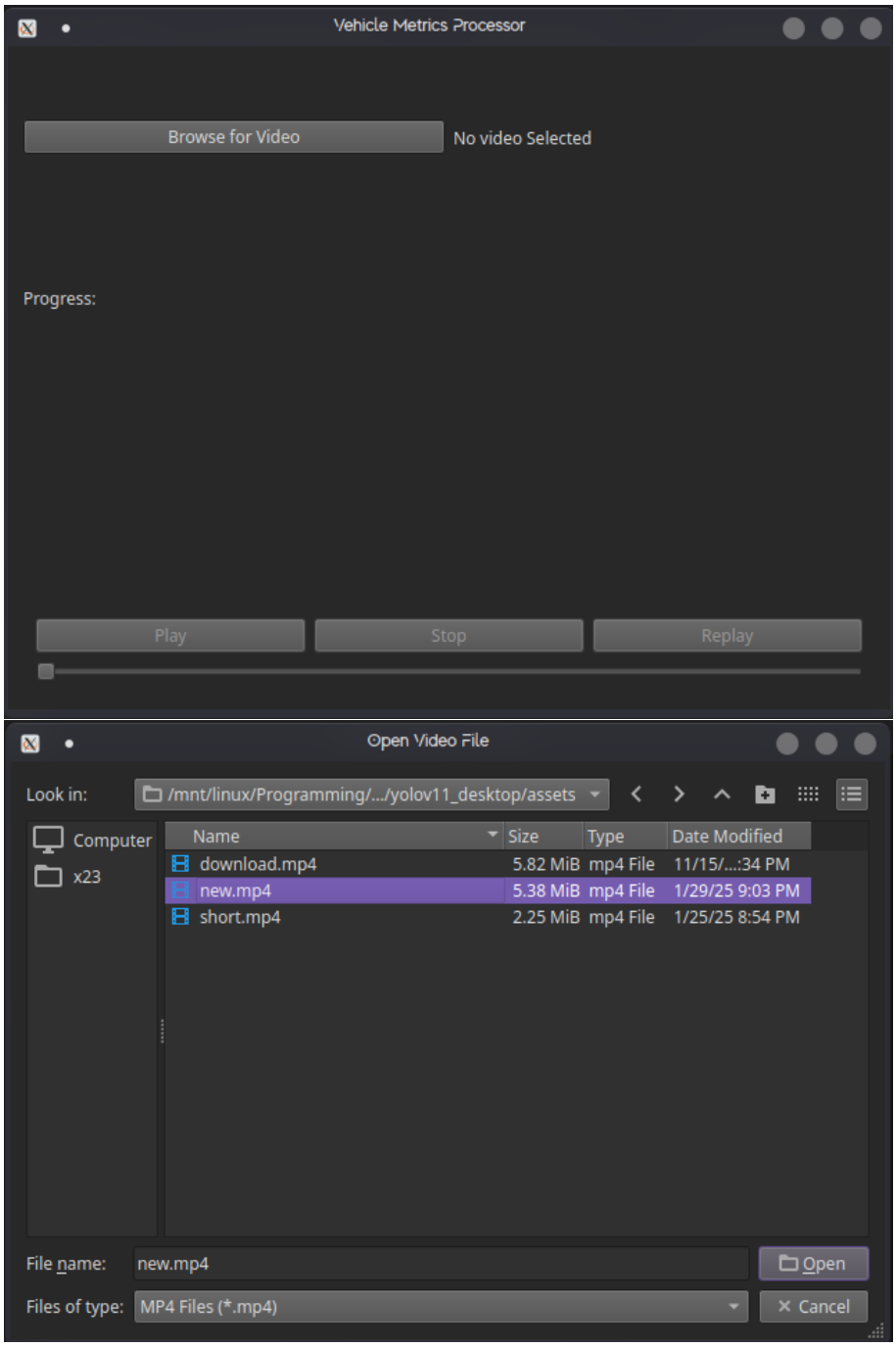

Figure 5.1: Selecting video from the GUI.

This eases the painstaking process of entering the video location everytime in the code. Moreover, it becomes easy for the user to understand what to do in spite of not having any prior programming experience.

Lane Adjustment

Upon selecting the video, the button for lane adjustment will appear.

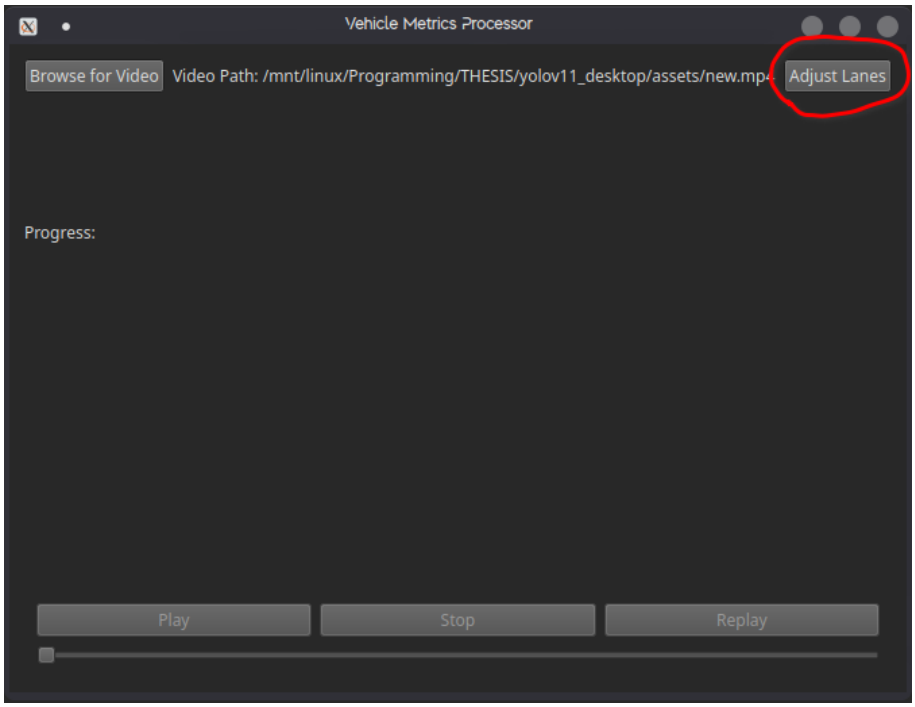

Figure 5.2: Lane Adjustment Button.

It will take us to a separate window, where we can add and position our separate in and out lanes. For now, there are 8 lanes in total, 4 for in and 4 for out. The lanes are basically line segments placed on the frame, which will be used as intersection references, and the lane number will be recorded.

Here, we will find checkboxes beside every lane, by checking or unchecking them, we can add or remove a lane. Each lane has 4 buttons, 2 for positioning and 2 for changing the length. And lastly, there is a label named Crossing, which is an invisible line. Any vehicle intersecting with this line is considered for data extraction and is then matched with the lanes on the screen. All the lanes are positioned over the crossing line. It can only be moved vertically, which means all lanes will also move with it for simplicity.

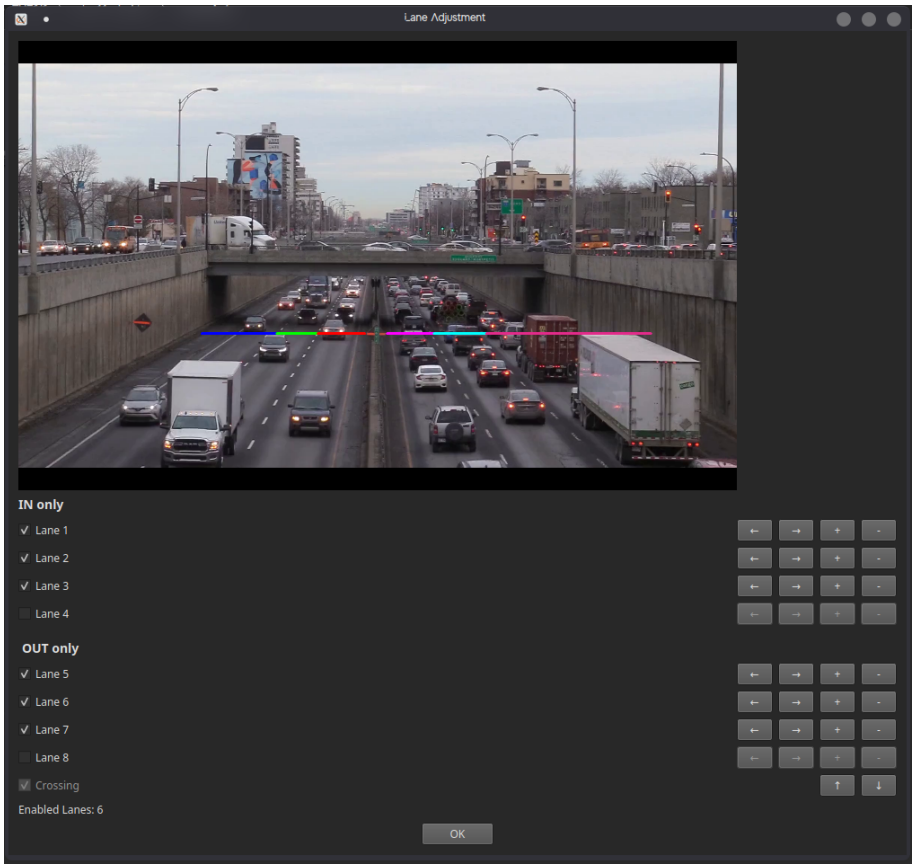

Figure 5.3: Lane Adjustment Window.

The purpose of adding such a feature to the software is that, in order to simulate traffic data in a 3D plane, we actually need a large number of attributes for every vehicle. In order to keep the software user friendly, we decided to go for only the crucial data for creating a simulation as close as possible to reality.

Starting the Extraction

After this point, it is a very straightforward job. We click on start, and we wait till the processing is finished. We will also be able to see the progress on the window.

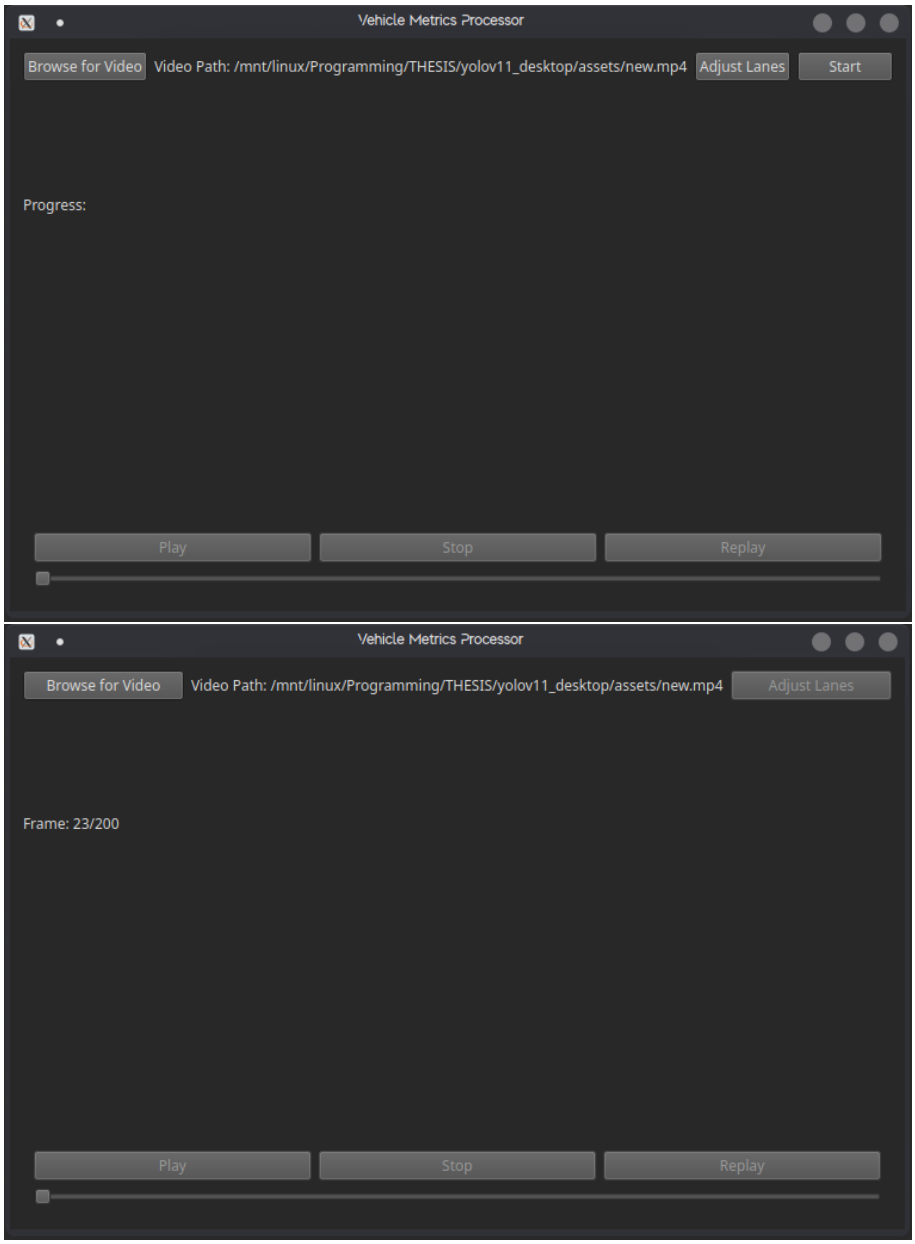

Figure 5.4: Starting the process.

Extraction Process

Each frame of the video is processed by our trained model. The model returns the bbox, tracking id of the vehicle and class data. The Bbox is stored in a list as a value in a dictionary against the vehicle class. This list is used to keep track of the previous position of the vehicle and therefore calculate an approximate velocity and direction. It is also used to draw a trail behind the vehicle in the annotated frame.

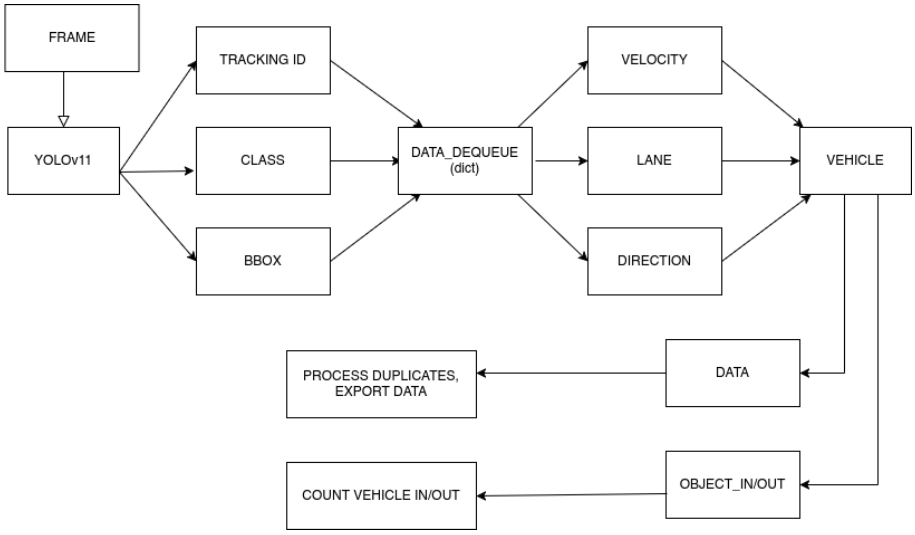

Figure 5.5: Extraction Flow Diagram.

A vehicle whose bbox has intersected with the crossing line is kept in multiple data structures in order to ensure that the categorization can stay as accurate as possible and also to ensure that no duplicate entry is exported in the text file.

5.1.3 SUMO

Selection and Transformation of Real-World Road Networks

For a realistic simulation environment, the first step is to select a real-world road network using OpenStreetMap (OSM), as shown in Figure 5.6. We used the Gulshan 02 intersection based on traffic density, intersection complexity, and relevance to real-world urban conditions in Bangladesh for our research. OSM provides highly detailed geospatial data, including road layouts, lane configurations, and intersections exported in .osm format.

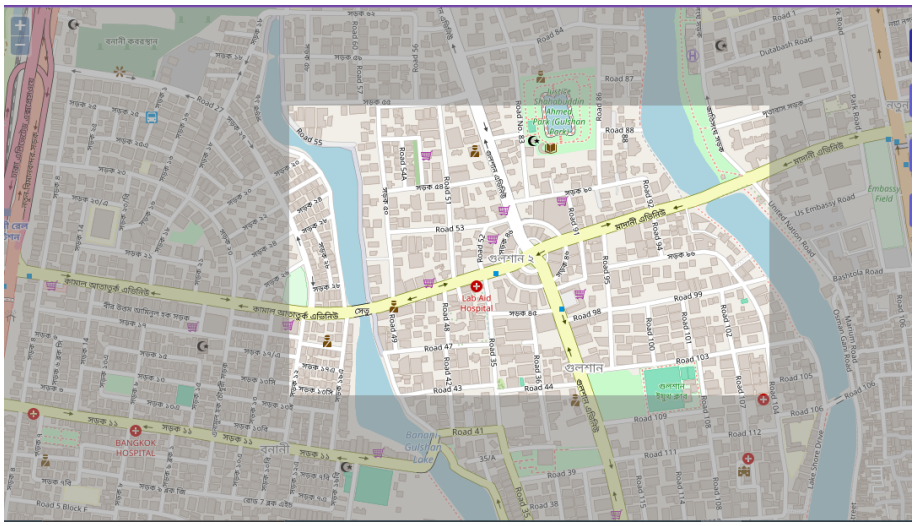

Figure 5.6: Area Map Selection from OSM

The raw OSM data is then converted into a structured SUMO road network (.net.xml) using the netconvert tool in SUMO:

```
netconvert --osm-files Gulshan02_Intersection.osm
--output-file Gulshan02_Intersection.net.xml
--roundabouts.guess --junctions.join
--tls.guess-signals --tls.discard-simple --tls.join
```

This process builds accurate road geometries, maintaining lane widths and segment lengths, automatic traffic light generation for controlled intersections and structured road hierarchy, and distinguishing between highways, arterial roads, and local streets. The network file then serves as the base infrastructure for traffic simulation, allowing vehicles to be introduced and monitored within SUMO's virtual environment. Figure 5.7 shows the SUMO road network of the Gulshan 02 intersection in comparison to its raw OSM visualization. The roads in the network are fragmented into edges, junctions, and lanes, each having unique IDs.

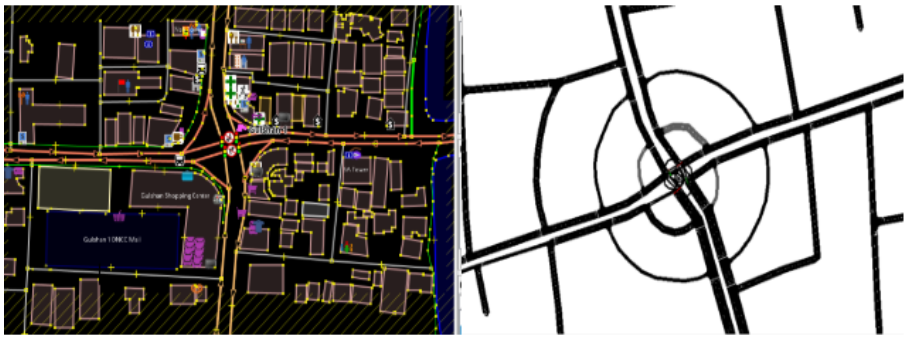

Figure 5.7: Raw OSM vs SUMO road network of Gulshan 02 intersection

Refinement and Calibration of the SUMO Road Network

To ensure real-world accuracy and proper traffic simulation, the road network is further edited manually using the SUMO netedit tool as shown in Figure 5.8. To prevent errors in simulation, SUMO's netcheck tool was used to identify disconnected road segments, overlapping lane connections, and so on. These were corrected using netedit, along with modifying signal timings to reflect real-world conditions, lane adjustments, including turn restrictions and lane-merging points, and intersection optimization.

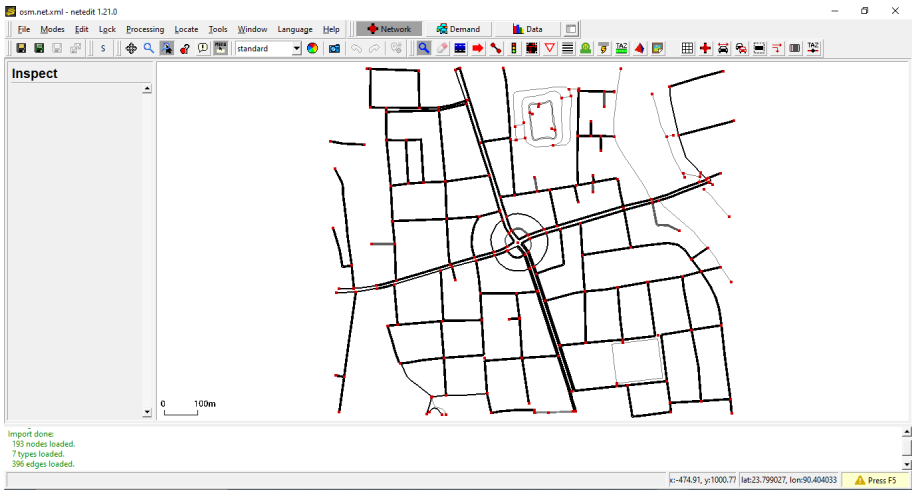

Figure 5.8: Netedit interface

These refinements resulted in a well-structured and error-free digital road network, forming the foundation for realistic traffic simulation.

Integrating Real-World Traffic Data from Object Detection

To incorporate real-time traffic flow dynamics, data from the YOLO-based object detection system was used, which detected vehicle count (traffic volume per unit time), vehicle categories (cars, buses, trucks, motorcycles) and lane-wise traffic distribution. The traffic data recorded in a txt file was then mapped into SUMO-compatible trips using a python script, generating a trips.xml file as illustrated in Figure 5.9. SUMO trips.xml file defines individual vehicle trips with departure times incremented per second, starting locations mapped to valid edges according to direction and destination edges estimated based on real-world routes.

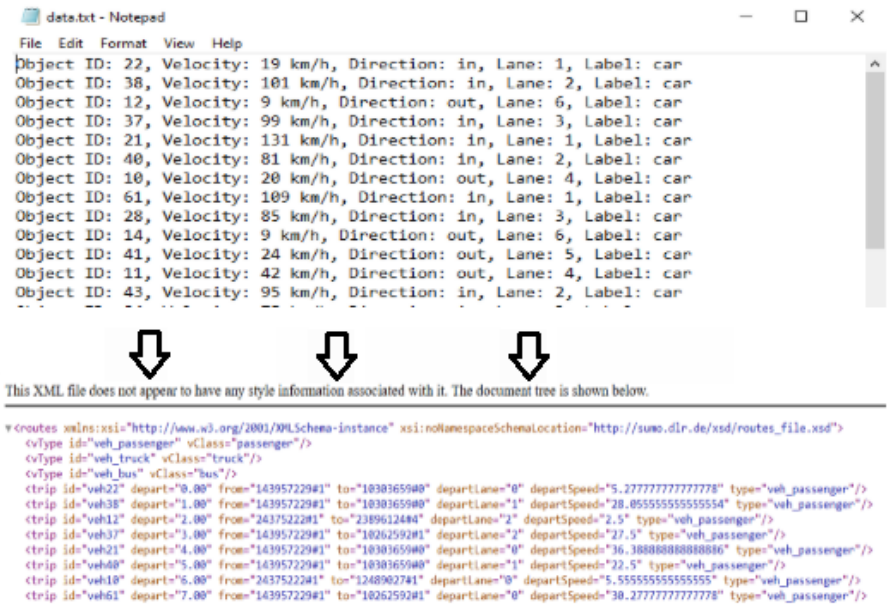

Figure 5.9: Vehicle detection output to trips.xml conversion

Route Generation and SUMO Simulation Execution

Once the trips file was generated, the next step involved assigning realistic routes to vehicles within the SUMO network. This is done by using SUMO's duarouter tool to compute optimized vehicle routes, considering real-world constraints since trips only provides starting and ending edges: **duarouter –net-file osm.net.xml –route-files yolo.trips.xml -o my_routes.rou.xml**
With the road network, routes, and traffic flow data in place, a `.sumocfg` configuration file is created with proper configurations, adding the network and route files and SUMO simulation is initiated using: **sumo-gui -c simulation_config.sumocfg**

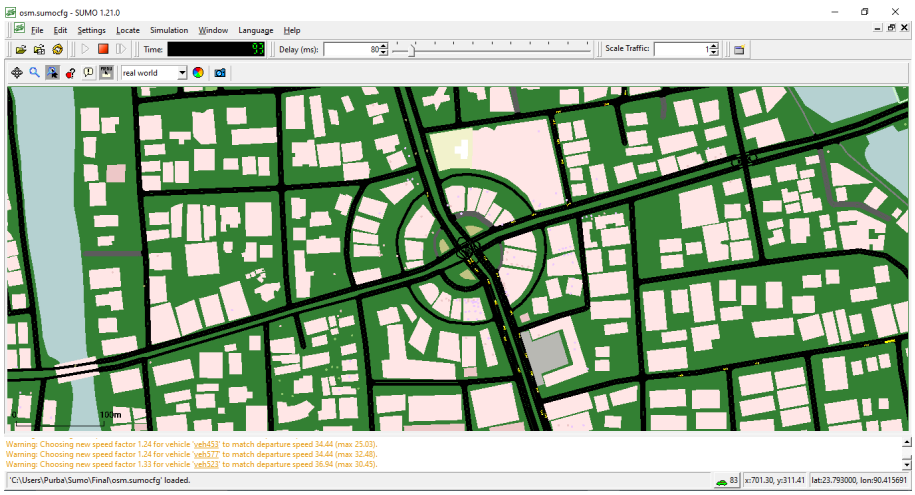

Figure 5.10: Traffic Simulation in SUMO Gui

Real-time Simulation and Modification Using TraCI

By connecting a python script to the Traffic Control Interface server port, the TraCI API is used to access the running simulation which is then manipulated dynamically to perform real-time modifications such as adding vehicles, retrieving vehicle information, adaptive traffic signal control and more. A summarized table is presented below:

Function	**Purpose**	**TraCI API Used**
Inject YOLO vehicles	Adds vehicles to simulation in real time	`traci.vehicle.add()`
Adjust Traffic Signals	Modifies traffic light phases dynamically	`traci.trafficlight.setPhase()`
Retrieve Vehicle Data	Monitors and collects data from vehicles in real time	`traci.vehicle.subscribe()`
Control Simulation Flow	Steps through SUMO in sync with traffic flow	`traci.simulationStep()`

Table 5.1: TraCI API Functions and Their Purpose

This setup provides an adaptive, data-driven approach to traffic control, improving simulation realism and accuracy.

5.1.4 Unity Setup

The method of using this form of visualization has its own set of versatility. The proper way of determining any problem requires a basis on which the planning can be done. This includes hard-coded AI characters that represent the environment accurately. This research describes an example of using AI to determine a possible solution to a basic problem. It will try to determine what combination of signal durations will give the best outcome on a busy street in Dhaka, Bangladesh. The full system will take the help of existing software to determine the proper enhancements of scenarios. This gives us the added benefit of being able to edit the Physics Engine (UNITY3D) to match our needs for better results.

Selecting Softwares

- **Blender**: For generating a 3D map of an area with the Blosm addon.
- **Unity**: For making the 3D map interactive, placing objects, and implementing movements.

Working Procedure

Blender
To complete this task, we used the Blosm addon for Blender. We need an API key from Google for using Google map data to use the add-on. In Blender, pressing

N will show the panel where we can select Blosm. Clicking on it will show us the following:

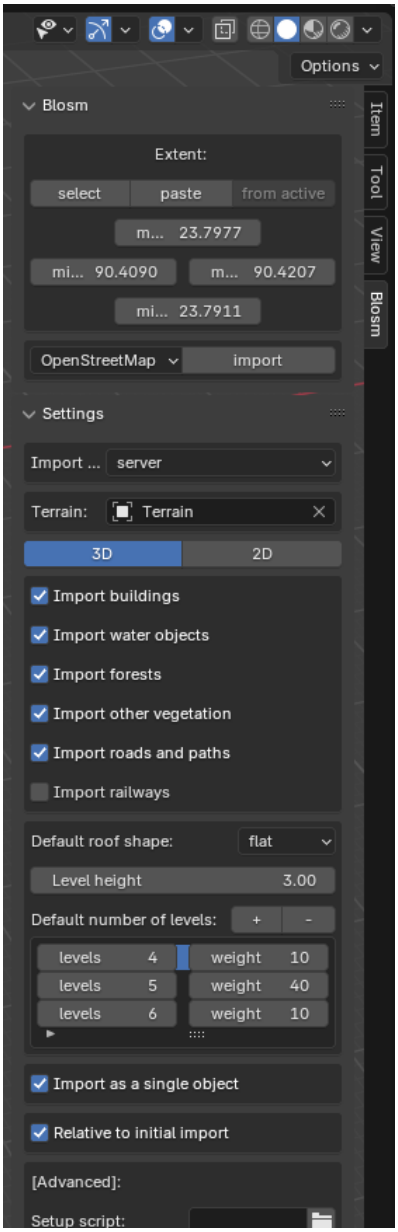

Figure 5.11: Blosm Addon.

Here, we click on select and it will take us to a website where we can select a rectangular map region according to our necessity.

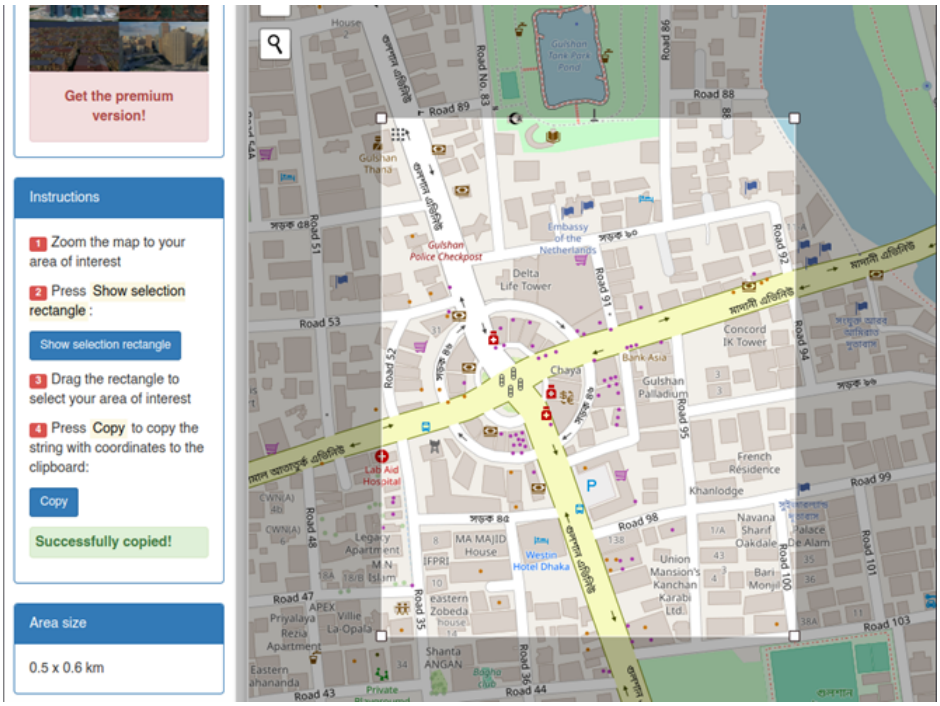

Figure 5.12: Map Selection for Blosm

In Blosm, we click on paste and the coordinates of this rectangular selection will be pasted. Then we import terrain first and then OpenStreetMap data and we will get a map like the following:

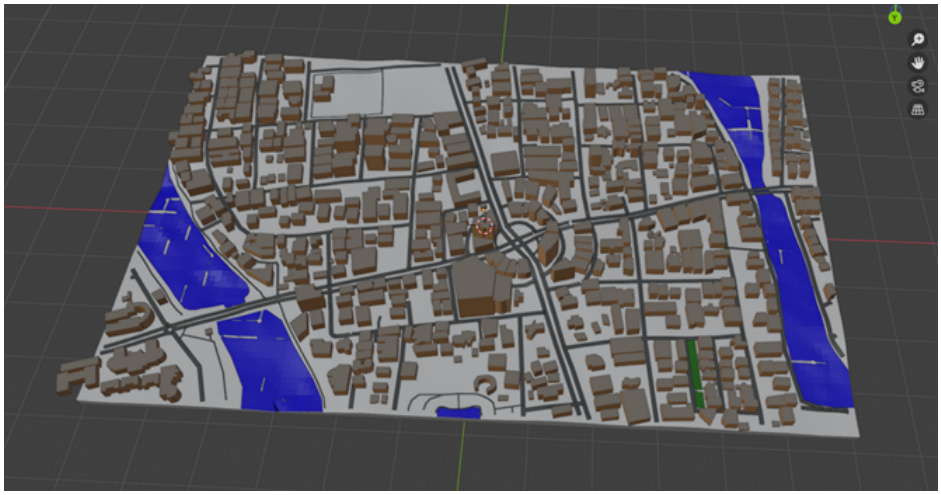

Figure 5.13: 3D Map Imported Using Blosm

We export it as .fbx which we will use in Unity3D software for further enhancements.

Digital Twin Setup

This part of the procedure varies depending on the intersection in focus. The objective of this part is to add realism to the scenarios being simulated. This may include dividers between lanes, props such as traffic cones or checkpoints, and disabling roads that have been modified in recent times but not updated in the Digital Twin model. Rather than just a setup, this step is an evolutionary process that will continuously be updated in the sequence of simulations that will follow. The analyst will have complete freedom of setting up the scenario as they please.

Unity3D

Once we import the previously created .fbx file in Unity3D, we will get the same view as we got in Blender.

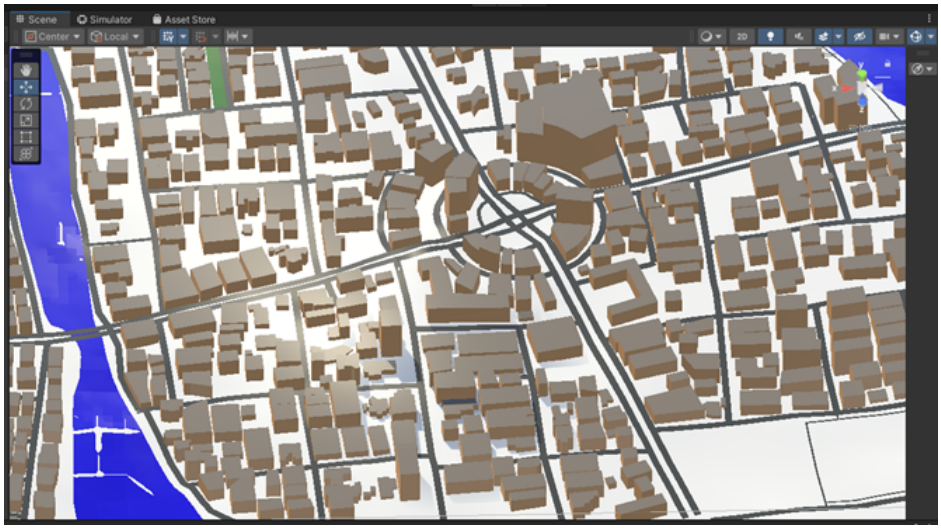

Figure 5.14: 3D map imported to Unity3D from Blender

Some roads may be divided due to map data discrepancy; we can join them by selecting them and pressing Ctrl + J. This digital twin can now be further updated to resemble a realistic environment. Other fine tunings, such as lighting, scaling, and object placement, can now be done before the waypoints are set and simulations are run.

Waypoint Navigation System

The process will start with the editor setting up multiple waypoints for the roads in the maps. These waypoints will help the cars pave the way to the intersection. These waypoints will mainly consist of an array of points, denoted by spheres, which the cars will move towards. These points will have a follower count list, which will count how many vehicles are following the point with some addition of waiting for the count to ensure even traffic distribution. The waiting count added to the followers list also helps analyze the most used lane in the simulation. One key part of waypoint communication is done by linking the waypoints together through an array of waypoints, similar to the concept of a linked list, yet different as it points

towards 3 directions rather than one. This part holds an imbalance of choice when a vehicle component reaches the waypoint. The vehicle will prefer to take the straight waypoint as its next destination 60% of the time, the values of which can be tweaked in the editor before simulations. Waypoints hold a key part in endpoints and intersections logic. A boolean variable labeled "isEndPoint" is responsible for destroying any vehicle objects pointing towards it. This is needed to destroy redundant objects that clutter up our simulations. Our main focus in the simulation is the intersection. Therefore, the objects heading outbound or leaving the main road are considered redundant or unnecessary. The endpoint flag is used to remove these extra objects from the simulation.

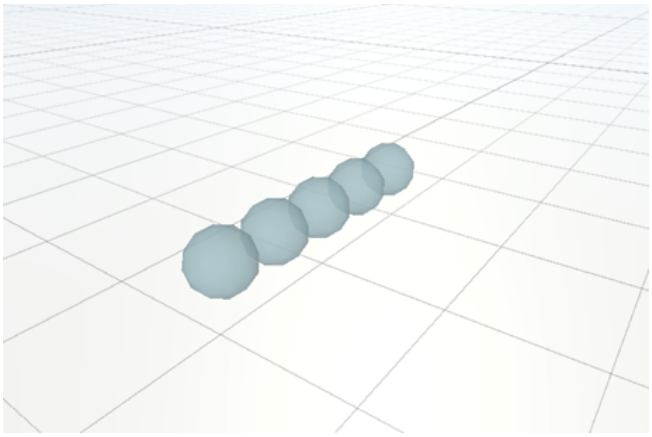

Figure 5.15: Waypoint Array representing the lane

Another flag named "isIntersection" is used to denote intersection waypoints. This stops any vehicles moving towards it and can change in real-time during the simulation to simulate the actions of red light and green light. The intersection point turns blue and tells the vehicles passing the points before it to stop as the intersection is ahead. The points before the intersection are labeled as entry-waypoints that turn green and red according to the intersection. These waypoints also hold alternative paths in the intersection. Rather than just going straight, the vehicle may change its path to another direction, such as left or right, adding realism to the simulations. The alternate paths are colored purple for visualization.

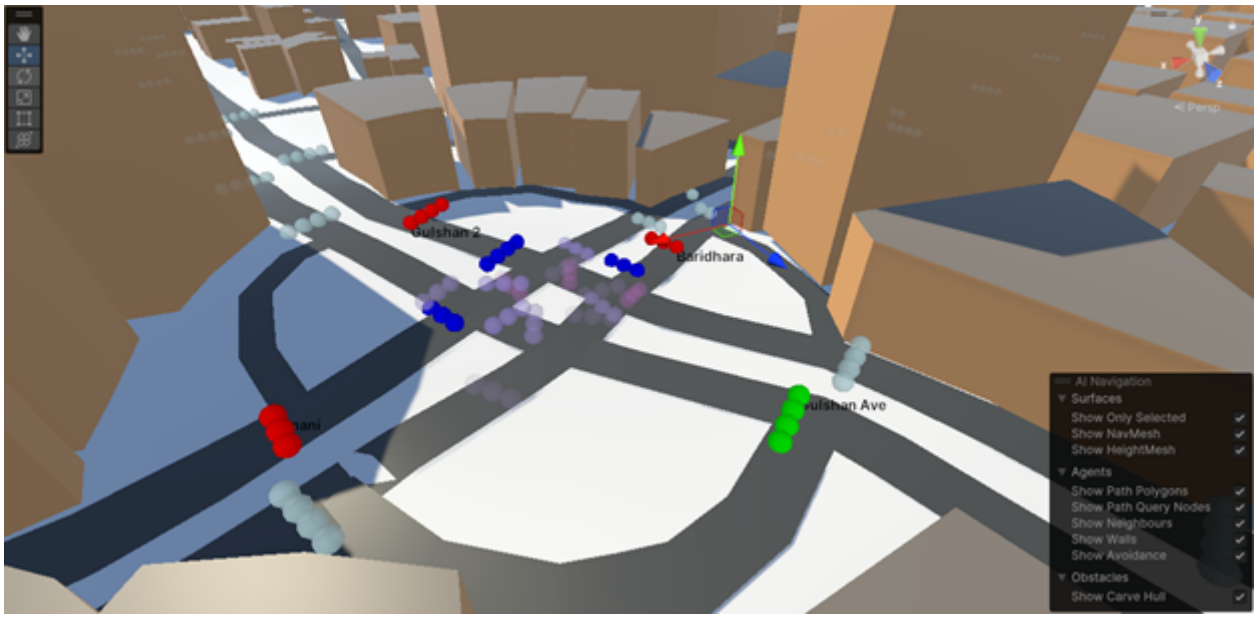

Figure 5.16: Light Signal visualization through waypoints

The waypoints are set along the roads, one side directed towards the intersection and one in the opposite direction. Rearranging these points and creating wave arrangements will simulate speed breakers and police checkpoints that will help further analyze how a car approaches an intersection, such as its speed and frequency. The paths influence the overall scenario of the intersection and determine the best path direction that the vehicles can take.

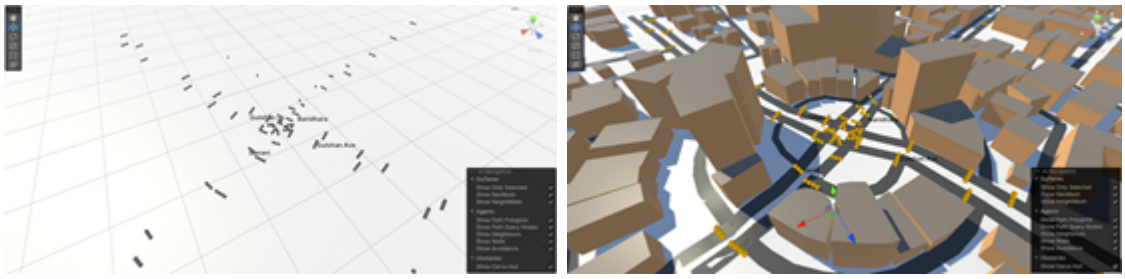

Figure 5.17: Wireframe of the waypoint setup

Vehicle Setup and Pathfinding System

The basics of the vehicle setup mainly have four components. The main body, bounding body, red block collider, and tracker sphere. The green body denotes the car itself for visualization purposes. The outer translucent bound mainly stores the physics of the vehicle as it is responsible for the mass, gravity, and collision detection. Other vehicle objects that come too close to the vehicle component will cause it to brake and wait for the cars in front to move. The red block collider triggers other objects behind the vehicle to stop. The bounding block collides with the red block to trigger the stop function. Lastly, the tracker sphere is the backbone of the vehicle logic. This invisible sphere is responsible for the movement and pathfinding of the object. The sphere works as a look-ahead tracker that moves a few units ahead of the vehicle. It then detects obstruction and changes direction instantly. This shift in direction is jagged yet undetectable in the simulations, as the sphere is transparent. The car then linearly interpolates (LERP) to the sphere, giving it smooth movement, which then leads the vehicle in the best possible direction for it to reach its destination waypoint.

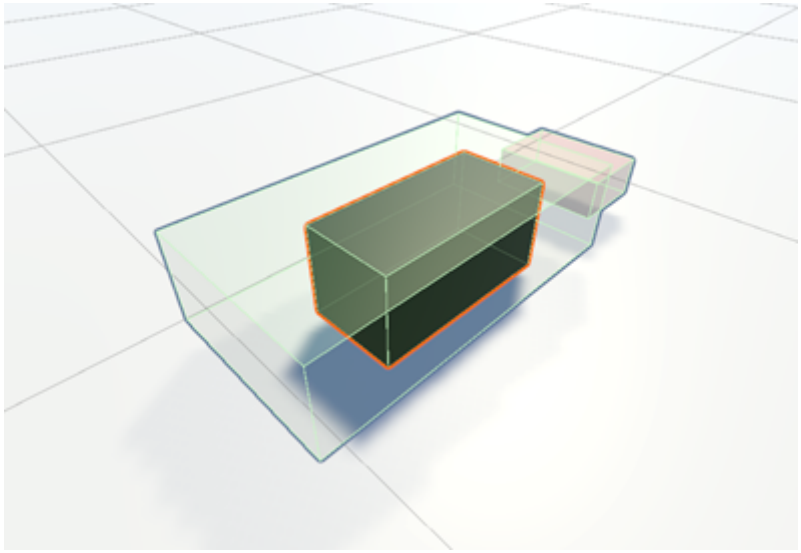

Figure 5.18: Car Setup

As shown previously, the waypoints hold a series of spheres with a follower count. When a vehicle is pursuing a waypoint, it looks for the sphere with the lowest follower count and follows the position of that sphere. Once the sphere is selected, it will

not change its destination until it reaches the destination. When the destination is reached, it creates a random integer with Unity's built-in random function and chooses the next waypoint from the array of the next waypoints. Straight denotes the direction towards the intersection and is selected mostly, while others have a lower probability of being chosen. The left and right waypoints always lead to an endpoint and the vehicle gets destroyed upon selecting these directions. The lanes as well as the mid lanes are considered in the spheres of the waypoints as Bangladesh has lenient lane restrictions.

Spawner System

The spawner is a point that spawns vehicles and gives them a particular speed and mass. These properties depend on the data detected by the YOLO Model.

Figure 5.19: Spawner point

The spawner uses the car speed and type of the results of the YOLO Model, and an average speed is determined for the different car types, which is then used when spawning. The data also creates a ratio of the different types. Finally, the mass is a manually set weight.

Intersection system and signal management

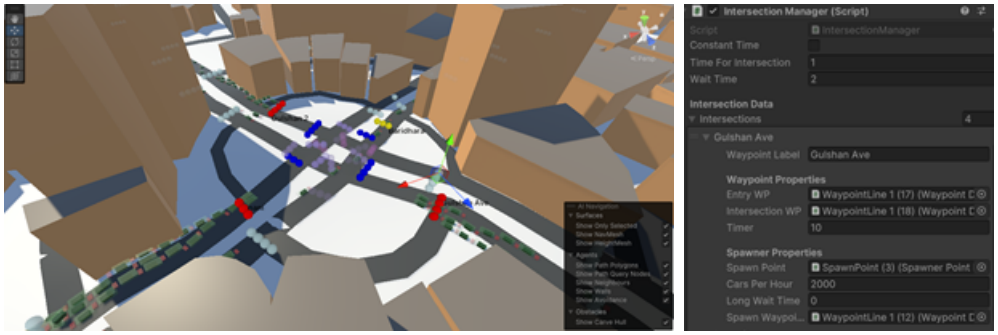

Figure 5.20: intersection system with signals

The intersection system mainly uses the coroutine function in Unity. This creates a timed function that runs the intersection signals. The intersection manager mainly takes the input of the time the signal will be set green, the waypoint with the intersection flag, the waypoint with the stop signal, and spawn details. Although the spawn points will receive YOLO data to generate ratios, the primary spawning frequency and intervals are set by the intersection manager. Alternate paths are also created and set in this section. The purple waypoints are set to ensure a smooth transition from one direction to another. It prevents vehicle collisions when entering other points in the intersection.

Analysis through Simulations

The analysis requires a series of simulation runs. We can divide these runs into different phases. The first phase will be labeled as the setup phase. The analysis will be unnecessary at this portion of the test runs. These runs will mainly consist of tweaks and fine-tuning of variables in the different components of the Unity setup. At this phase, the waypoints are adjusted, the vehicle waypoint detection radius is tweaked, the tracker sphere is adjusted, and the number of lanes is checked. Other settings, such as timer settings, cars per hour value, and other parameters, are set and noted, as these values will be further improved as we continue our analysis. The second phase is considered the most realistic, as these runs will use calculated values to run simulations. The values are: the ratio of types of vehicles, cars per hour, and spawn intervals.

$$\text{Cars Per Hour} = \left(\frac{\text{total cars in result}}{\text{total time of recording in mins}} \right) \times 60$$

$$\text{Spawn Intervals} = \frac{3600}{\text{Cars Per Hour}}$$

After running a series of test runs, we should note the traffic intensity of each entry.

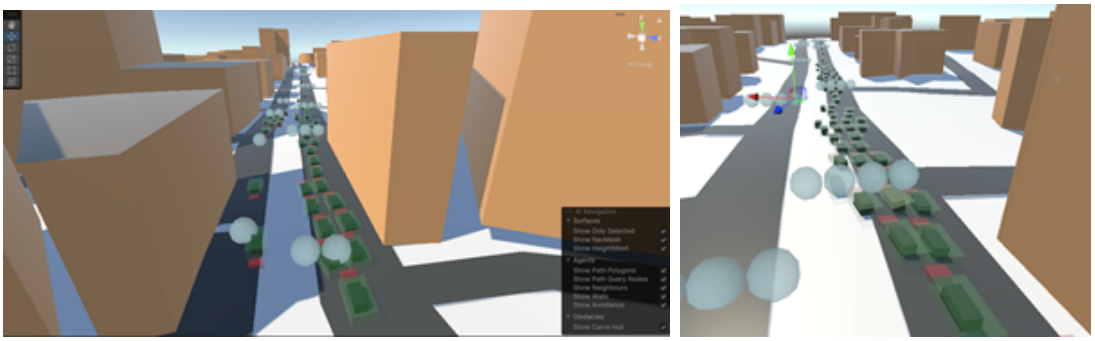

Figure 5.21: Comparison of Congestion of an entry lane

Given the setup phase and parameters are set properly, the congestion choke points can be easily detected in the editor. Some common signs are overflowing traffic, glitching, and fighting vehicles that show the congestion rate of that particular entry needs updating. The points are then further examined in the real world to relate to

our simulated environment. After identifying potential problematic areas, we must make changes to waypoints, timer variables, and the order of signal change to solve the choke holds of the intersection. These changes can then further be experimented with in the real world. The last phase is labeled as benchmarking. In this phase, we must override the calculated data and provide a worst-case scenario to check if solutions hold. This can be used to determine a foolproof plan. Note that when increasing traffic, we must use the realistic ratio of cars per hour values for each entry. For example, if entry 1 is a secured area with limited traffic and entry 2 is an entry to the central business district, the extrapolated value of entry 1 can never be greater than entry 2.

Calibration of the intersections

The values of time for each side must be tweaked and simulations must be run for each calibration to determine the best combination of values for the intersection to have the best efficiency. This process of calibration can be done manually or through various Transport Engineering Concepts. All formulas can easily be converted as the system is normalized to real-life values.

5.2 Experimental Findings

5.2.1 YOLO Model Results

The evaluation of the YOLOv11 model for vehicle detection is presented in this section, analyzing its overall performance as well as class-specific metrics. The findings are based on precision, recall, and mean average precision (mAP) at IoU thresholds of 0.5 (mAP50) and 0.5-0.95 (mAP50-95). These metrics provide insight into the model's accuracy and reliability in detecting different vehicle classes.

Overall Model Performance

The YOLOv11 model was tested on a dataset containing 889 images and 2,758 labeled object instances. The model achieved a precision of 91.8%, meaning that 91.8% of detected objects were correctly classified. Its recall stood at 90%, indicating that 90% of actual instances were successfully detected.

Class-Specific Performance

The mAP50 score of 94.8% suggests strong detection capability at a 0.5 IoU threshold, while the mAP50-95 score of 72.2% indicates moderate performance when stricter localization requirements are applied.

Class	Images	Instances	Precision	Recall	mAP50	mAP50-95
All	889	2758	0.918	0.900	0.948	0.722
Bicycle	95	197	0.937	0.910	0.975	0.677
Bike	150	206	0.925	0.899	0.962	0.711
Boat	76	344	0.894	0.811	0.890	0.563
Bus	98	171	0.879	0.895	0.927	0.704
Car	206	368	0.899	0.875	0.938	0.705
CNG	114	187	0.977	0.947	0.978	0.793
Easybike	96	192	0.966	0.888	0.948	0.752
Horsecart	27	33	0.763	0.975	0.916	0.726
Launch	79	154	0.912	0.818	0.927	0.670
Leguna	28	71	0.959	0.958	0.986	0.852
Rickshaw	161	374	0.874	0.821	0.894	0.623
Tractor	43	51	0.961	0.955	0.976	0.860
Truck	95	144	0.962	0.883	0.955	0.723
Van	118	207	0.940	0.908	0.968	0.764
Wheelbarrow	42	59	0.921	0.949	0.977	0.708

Table 5.2: Performance metrics for different vehicle classes.

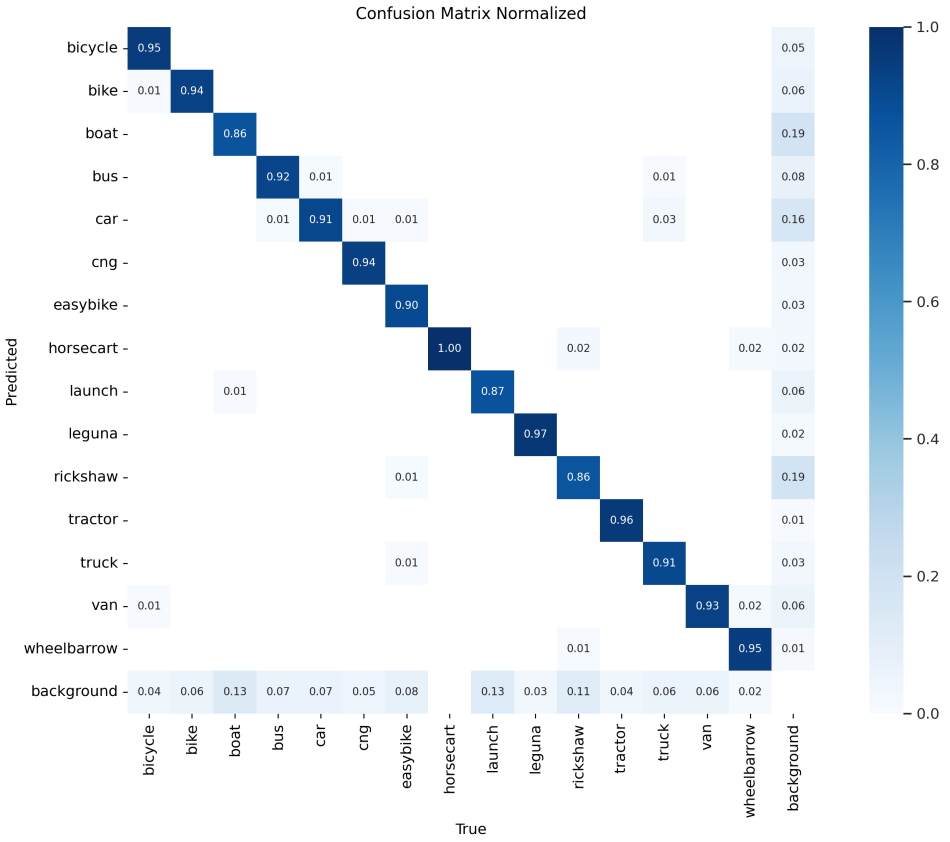

Figure 5.22: Confusion matrix of our fine-tuned YOLOv11 vehicle detection model for the test set

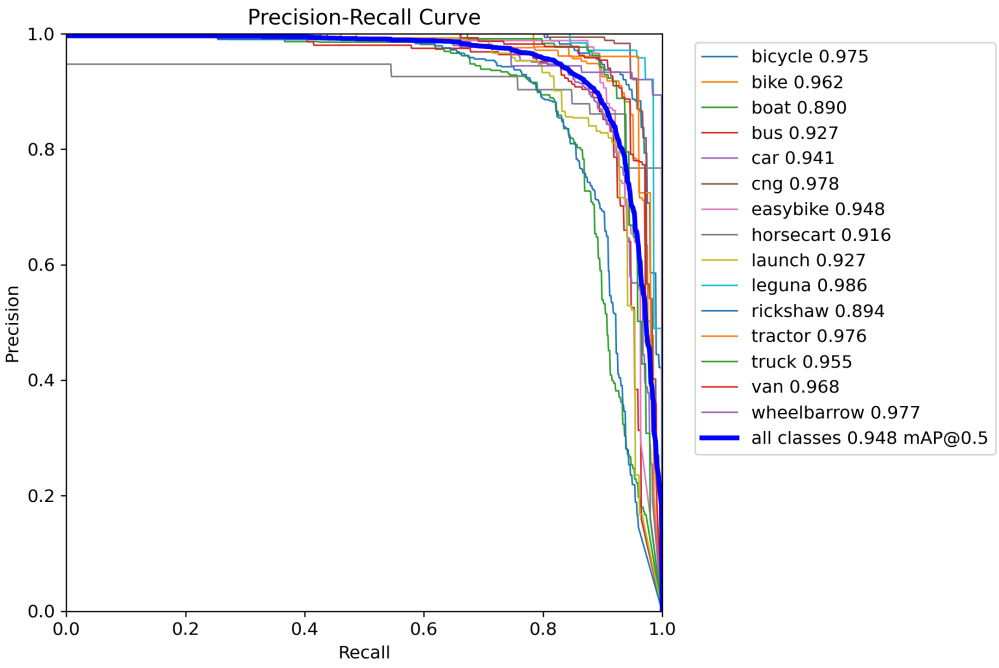

Figure 5.23: Precision-recall diagram of our model on the test set.

Table 5.2 illustrates the model's performance on the test set for various vehicle categories. The CNG (Auto-rickshaw) class achieved the highest precision (97.7%) and recall (94.7%), indicating a high rate of accurate detections with minimal false positives and false negatives. The Car, Bus, Truck, and Van categories also displayed strong detection results, with precision and recall values between 88% and 90%. Although these categories were generally well-identified, there were occasional misclassifications or missed detections. On the other hand, the Boat category had the lowest performance (mAP50-95: 56.3%), highlighting difficulties in detection. The confusion matrix in Figure 5.22 reveals that the background was often misclassified as boats. This could be attributed to the reflection of boats on the water being erroneously identified as boats.

5.2.2 Extracted Data

After the detection and tracking procedure is finished, the annotated video will be played in the window. In the annotation, we will be able to see the bounding boxes (bbox) of the vehicles, their ID numbers, velocity, and a count of the number of vehicles entering and leaving the frame with respect to the crossing line.

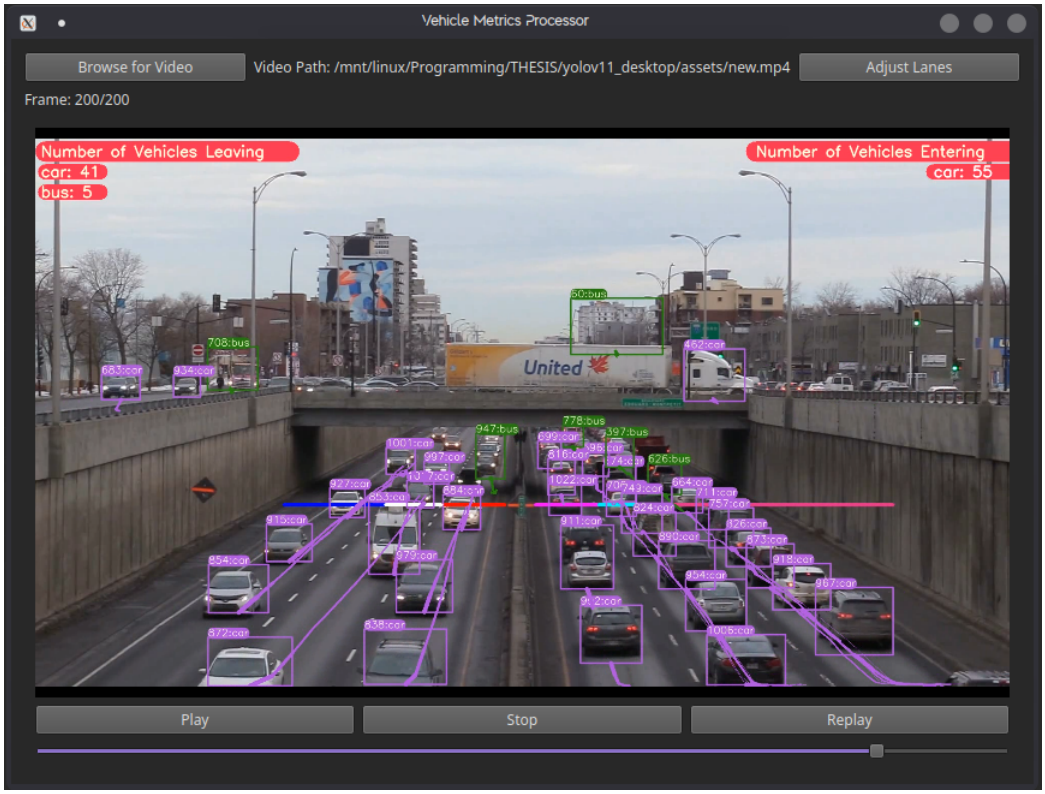

Figure 5.24: Annotated video

As soon as the video processing completes, a "data.txt" file will be generated, which will contain the records of the vehicles intersecting the crossing line, containing their velocity, direction, lane, and type.

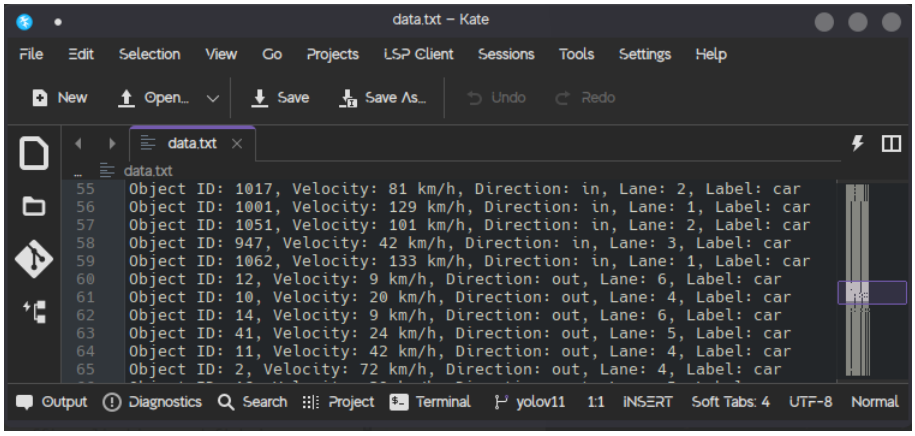

Figure 5.25: Exported Data

5.2.3 SUMO Simulation Analysis

Using SUMO and TraCI, we extracted a variety of traffic-related data, including vehicle data containing speed, lane position, route, acceleration, waiting time, and lane data containing average speed of vehicles on a lane, vehicle count, lane occupancy, and more. These extracted data points allow for a detailed analysis of simulated traffic conditions, making it possible to compare SUMO's rule-based simulation with real-world traffic scenarios in Bangladesh. The traffic counts in different roads and lanes along with other information from the lane data can further be used to identify busy roads and congestions and used in traffic management optimization.


```
Step 47: veh290 -> {'speed': 13.143230344744302, 'position': (619.3434713186668, 409.0328321287606), 'acceleration': -2.362305258649897, 'lane':
'24375222#4_2', 'route': ('24375222#1', '24375222#2', '24375222#3', '24375222#4', '23895139', '23896124#0', '23896124#1', '23896124#2', '23896124#3', '23896124#4'),
'co2_emission': 0.0, 'fuel_consumption': 0.0, 'waiting_time': 0.0}

Step 47: veh297 -> {'speed': 26.220220866559384, 'position': (630.421659384931, 321.1609148583869), 'acceleration': 0.5062680938327375, 'lane':
'143957229#2_1', 'route': ('143957229#1', '143957229#2', '143957229#3', '143957229#4', '143957229#5', '498169188#0', '498169188#1', '498169188#2', '498169188#3',
'498169188#4', '498169188#5', '498169188#6', '-681506963#3', '-681506963#2', '10303658', '10303659#0'), 'co2_emission': 8079.732988473502, 'fuel_consumption':
2577.043938829016, 'waiting_time': 2.0}

Step 47: veh301 -> {'speed': 26.624441280418317, 'position': (619.9426963582893, 354.83855153862186), 'acceleration': -0.762779293535278, 'lane': '143957229#2_1',
'route': ('143957229#1', '143957229#2', '143957229#3', '143957229#4', '143957229#5', '498169188#0', '498169188#1', '498169188#2', '498169188#3', '498169188#4',
'498169188#5', '498169188#6', '-681506963#3', '-681506963#2', '10303658', '10303659#0'), 'co2_emission': 0.0, 'fuel_consumption': 0.0, 'waiting_time': 3.0}

Step 47: veh325 -> {'speed': 18.57627778716202, 'position': (602.1776978246982, 418.3028907778534), 'acceleration': 2.1217023920267835, 'lane':
'143957229#3_0', 'route': ('143957229#1', '143957229#2', '143957229#3', '143957229#4', '143957229#5', '1088038754#0', '1088038754#1', '11075217#0', '10262593',
'-10262594#4', '10262590#0', '10262592#0', '10262592#1'), 'co2_emission': 13197.055375778207, 'fuel_consumption': 4209.188650715727, 'waiting_time': 0.0}

Step 47: veh33 -> {'speed': 10.61128292385329, 'position': (739.7843609873247, 367.14094319419695), 'acceleration': -0.42994719476619814, 'lane': '23895139_0',
'route': ('24375222#1', '24375222#2', '24375222#3', '24375222#4', '23895139', '23896124#0', '23896124#1', '23896124#2', '23896124#3', '23896124#4'),
'co2_emission': 0.0, 'fuel_consumption': 0.0, 'waiting_time': 0.0}

Step 47: veh331 -> {'speed': 5.56691026448542, 'position': (594.8532372341575, 429.7351253961068), 'acceleration': -4.5, 'lane': '143957229#3_1', 'route':
('143957229#1', '143957229#2', '143957229#3', '143957229#4', '143957229#5', '498169188#0', '498169188#1', '498169188#2', '498169188#3', '498169188#4', '498169188#5',
'498169188#6', '-681506963#3', '-681506963#2', '10303658', '10303659#0'), 'co2_emission': 0.0, 'fuel_consumption': 0.0, 'waiting_time': 0.0}

Step 47: veh336 -> {'speed': 15.6191648663151, 'position': (588.2311098395769, 486.6521256533545), 'acceleration': 2.0387826983584105, 'lane': ':415937616_2_1',
'route': ('24375222#1', '24375222#2', '24375222#3', '24375222#4', '23895139', '23896124#0', '23896124#1', '23896124#2', '23896124#3', '23896124#4'), 'co2_emission':
10813.899315832094, 'fuel_consumption': 3449.088533576553, 'waiting_time': 0.0}

Step 47: veh361 -> {'speed': 27.029436894808896, 'position': (700.0226643307167, 108.84753498787157), 'acceleration': 2.029436894808896, 'lane': '143957229#1_1',
'route': ('143957229#1', '143957229#2', '143957229#3', '143957229#4', '143957229#5', '498169188#0', '498169188#1', '498169188#2', '498169188#3', '498169188#4',
'498169188#5', '498169188#6', '-681506963#3', '-681506963#2', '10303658', '10303659#0'), 'co2_emission': 19151.537062944506, 'fuel_consumption': 6108.366407579309,
'waiting_time': 0.0}
```


Figure 5.26: Extracted vehicle data

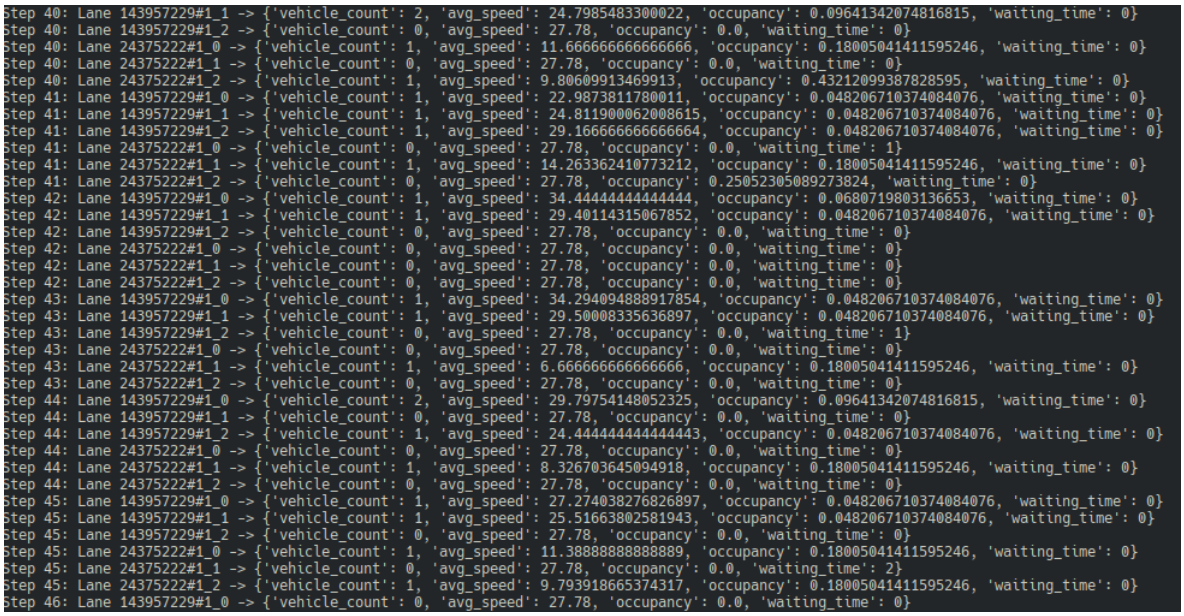

Figure 5.27: Extracted Lane data

Furthermore, through the simulation, it was observed that SUMO's structured, rule-based traffic flow resulted in smoother movement and significantly reduced congestion compared to real-world urban traffic conditions in Bangladesh. In contrast, real-world traffic in Bangladesh is unpredictable and unstructured, leading to frequent congestions and inefficient flow. The simulation demonstrated that SUMO could efficiently manage the same volume of traffic while reducing congestion, which on the other hand, would cause bottlenecks in the roads of Dhaka due to frequent lane and traffic signal violations and haphazard rules. The findings suggest that Bangladesh's traffic congestion issues could be mitigated through structured enforcement and proper implementation of strict traffic rrules, as demonstrated by SUMO.

5.2.4 Qualitative Analysis of Unity

The analytical results of the Unity Simulation hold more qualitative value than quantitative. Success is subjective, and the validity of the results can only be confirmed after analyzing their implementation in the real world. Our paper presents a simpler methodology for analyzing traffic in Bangladesh. Therefore, the findings will show the theoretical output of the methodology discussed in the paper and a basic analysis of the outcomes achieved in our simulation.

Phase One Outcomes

This phase targets updating waypoints and other minuscule details to make the simulation resemble the real world. For example, if vehicles are moving through buildings, it becomes unrealistic. The Figure 5.28 shows a proper setup of the waypoints with visual traffic moving as intended.

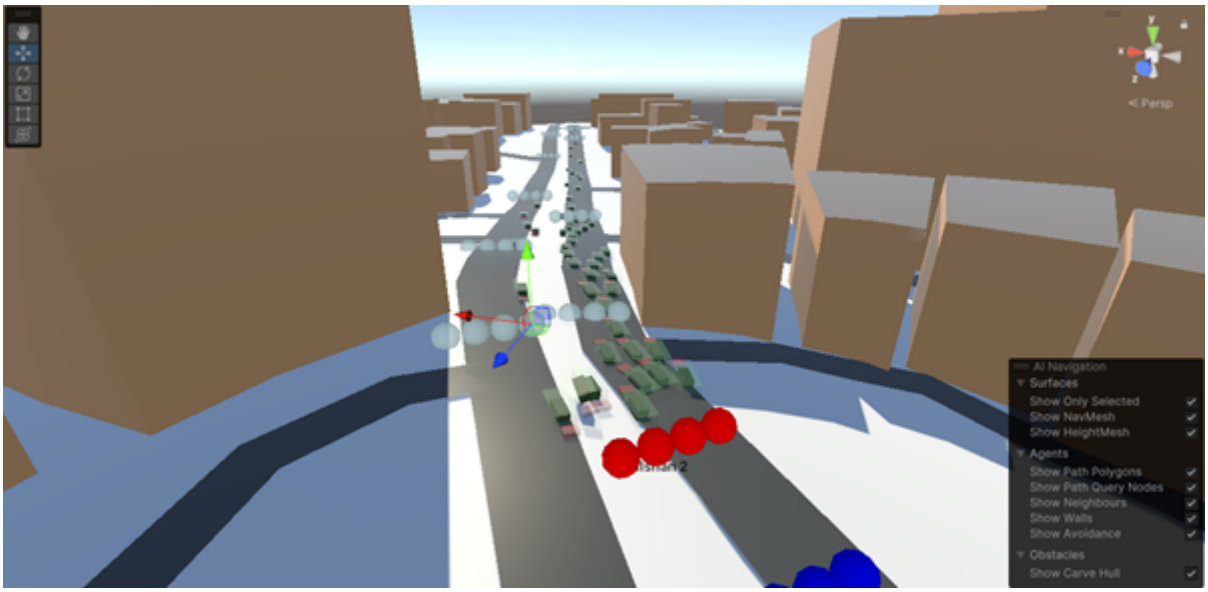

Figure 5.28: Tweaked Waypoint setup

Phase Two Outcomes

The simulation is run with average values for each entry point. The simulation is kept at a constant time per entry point of the intersection and is observed to have more overall traffic from the Banani side entry. The Figure 5.29 shows entries with less traffic in other points compared to Banani. This stage of simulations uses values from the YOLO models in spawners to create realistic spawning mechanisms.

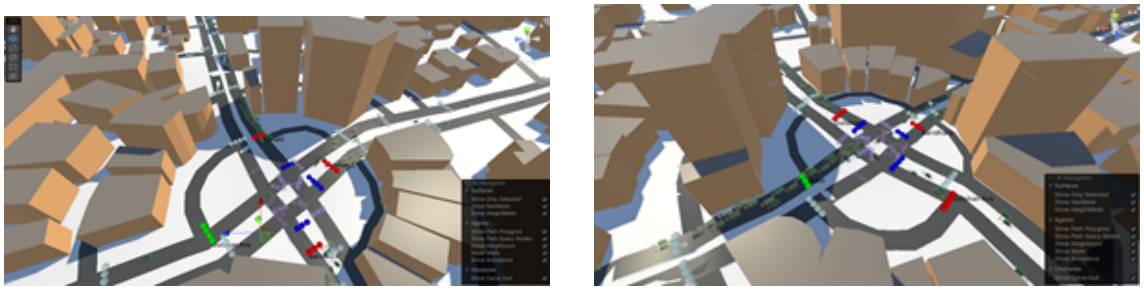

Figure 5.29: before and after phase two

The values and ratios calculated are manually set in each spawner. These values can be changed to refine the spawning and solve inaccuracies of the achieved data.

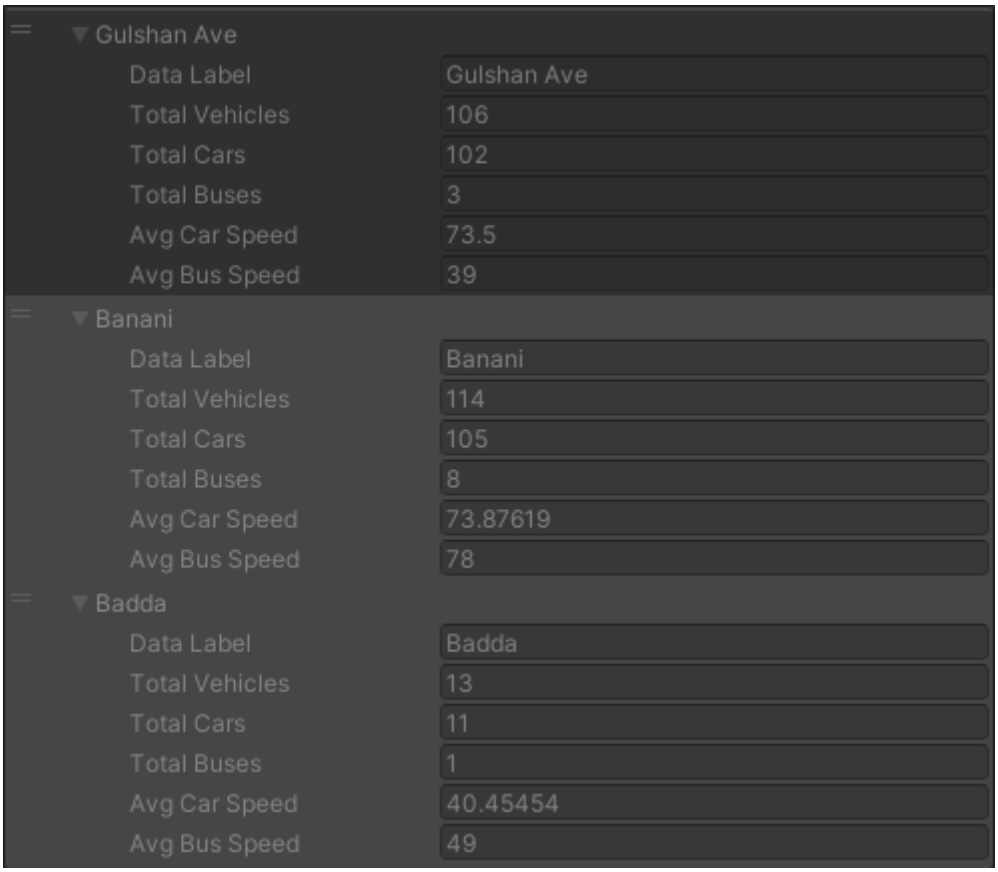

Figure 5.30: Data extracted metrics

Phase Three Outcomes

This phase uses unrealistic values to check if the intersection properties set in the previous phases are feasible. Here, we ensured the ratio of cars per hour from the previous simulations remains intact, but the values themselves are proportionally increased.

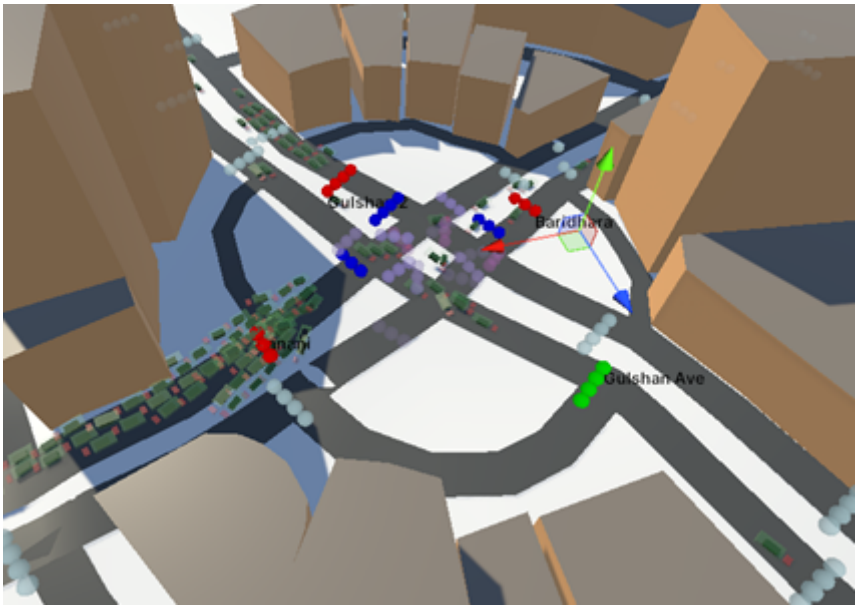

Figure 5.31: Overflow of traffic after upscaling in phase three

Chapter 6

Discussion

The trained model has demonstrated sufficient effectiveness in achieving our research objective of extracting traffic data, highlighting the crucial role of vehicle detection in analyzing road traffic patterns. The use of transfer learning significantly reduced development time while improving tracking and tracing capabilities. When compared with other vehicle detection models, our approach achieved the highest mean average precision (mAP) on the Poribohon-BD [16] dataset. The only model that surpassed our mAP score was the Custom PMB [47] dataset, which was specifically trained focusing on 8 classes of vehicles from their own developed custom dataset relevant to the Padma Multipurpose Bridge only. In Table 6.1, an asterisk (*) has been used in the #Param field where the number of parameters was not provided in the original reference but was retrieved from the official YOLO documentation.

Ref.	**Dataset**	**#Classes**	**Model**	**#Param**	**mAP**
[Ours]	P-BD	15	YOLOv11m	20.1 M	0.948
[78]	P-BD	15	YOLOv8S	11.2 M	0.913
[53]	D-AI	21	YOLOv7	36.9 M	0.512
[47]	CUSTOM PMB	8	YOLOv7X	71.3 M	0.968
[56]	CUSTOM	20	YOLOv7	37.2 M	0.802
[23]	D-AI	21	YOLOv5X	86.7 M	0.458
[37]	CUSTOM	21	YOLOv5L	71.3 M	0.968
[27]	D-AI+P-BD	21	YOLOv5L	87.7 M	0..287
[19]	D-AI	21	YOLOv5L+X	-	0.895
[29]	D-AI	21	YOLOv3	65.3 M	0.76

Table 6.1: Comparison of our proposed fine-tuned YOLOv11m model with previous works on Dhaka (or Bangladesh) based dataset. [66]

Since our primary focus is on traffic congestion, vehicle types such as boats and launches were not highly relevant to this study. A more refined dataset that prioritizes vehicles commonly found on Bangladeshi roads would have been more suitable. Additionally, a significant portion of the dataset consisted of augmented images, which may have led to an overestimation of the model's performance. When splitting the dataset into training, validation, and test sets, some augmented images from the training set may have been included in the test set, potentially skewing

the results. [66] observed that testing their model on independently collected video data highlighted the need for further validation and reinforced the importance of a more standardized dataset. However, the limited availability of datasets meeting our specific requirements, along with the Poribohon-BD dataset remaining unchanged since its initial release, highlights the pressing need for an updated dataset.

Despite the model's high accuracy, certain limitations were observed. It occasionally misclassified objects and struggled to detect vehicles that were either too small or too distant. Additionally, velocity estimation relied on the Euclidean Distance formula, leading to inconsistencies where vehicles were assigned unrealistically high or low speeds in the exported data. This issue became particularly evident when analyzing stock videos at increased playback speeds, which often resulted in exaggerated velocity values.

These findings highlight the strengths and weaknesses of our approach. While the model effectively contributes to traffic data analysis, further improvements in dataset quality and velocity estimation methods are necessary to enhance its overall reliability. Future work should focus on updating the dataset with more representative vehicle categories and exploring alternative velocity estimation techniques to minimize inaccuracies.

Following vehicle detection using YOLOv11, the extracted traffic data was integrated into SUMO for 2D traffic simulation. The decision to use SUMO for simulation was driven by its ability to model urban traffic flows using real-world networks and controlled traffic rules. However, preprocessing steps were required to bridge the gap between the extracted traffic data and SUMO simulations, converting detected traffic data into SUMO-compatible XML format. The data contained vehicles, including non-motorized and three-wheeler vehicles that are prevalent in Bangladesh, such as rickshaws and CNG auto-rickshaws, which SUMO could not model and so were replaced with cycles and motorbikes. Although the absence of these vehicle types in the simulation limits the realism of the results to some extent, the SUMO simulation demonstrated some key insights, such as the challenge in modeling real-world Bangladesh traffic, the identification of high congestion points, and the potential outcome of rule-based traffic control in SUMO implemented in real life.

Our experimental simulations done in unity suggest that we have achieved a relevant and realistic simulation. Minor adjustments and enhancements are still required to improve accuracy; however, overall, the simulation effectively conveys a sense of realism.

Uneven traffic distribution at different entry points highlighted the need for better timing mechanisms at intersections. The Banani entry experienced the highest congestion, according to our simulation, suggesting that factors such as road width, signal timing, and vehicle arrival rates should be further analyzed. After updating traffic waiting times, the traffic congestion showed a better traffic distribution at the intersection. This emphasizes the importance of adaptive timing strategies in traffic management. Our experimentation revealed that tweaking the values did not fully resolve the congestion problem, indicating further timing adjustments are necessary

for fine-tuning.

These simulations are similar to the previous research done in [77][74] yet differ in practice. In most research, this method of SUMO and Unity is used to create a simulative environment for an immersive experience, such as driving simulators or network analysis. Our methodology shows more of a trail of analysis to determine a plausible solution to congestion in particular intersections. Simulations created in major industrial software such as Matlab's Roadrunner, although it has more moving parts and better physics implementations, lack the behavioral patterns of Bangladeshi drivers. The aggressive nature and random lane selection with no clear lane division are problems that most conventional simulation software will ignore. The simulations done in this research, show a feasible and cost-efficient technique to dissect traffic-related issues in Bangladesh.

The discussion effectively addresses the research objectives by demonstrating how the developed digital twin integrates real-world traffic data with SUMO and Unity to simulate Dhaka's traffic conditions. It assists urban traffic analysis by identifying congestion points, particularly highlighting Banani as a critical bottleneck, and tailoring the simulation to Bangladeshi driving behaviors, which differ from conventional structured traffic models. The study also benchmarks infrastructure capacity by simulating real-time traffic scenarios, though its reliance on pre-recorded footage limits adaptability, necessitating future integration of live traffic data. Additionally, the research evaluates speculative traffic regulations, revealing the need for improved signal timing but also identifying limitations in velocity estimation accuracy. While the study establishes a foundation for a sandbox toolkit enabling urban analysts to experiment with traffic rules and lane usage, further improvements in visualization, dataset quality, and real-time adaptability are required for broader practical applications in urban planning.

Chapter 7

Limitation and Future Work

While the research successfully developed a methodology for a digital twin based traffic management system for Bangladesh using YOLOv11, SUMO, and Unity, several limitations were encountered that impacted the accuracy and applicability of the study. These limitations primarily arise from model limitations, data availability, real-world constraints, and technical challenges in modeling Dhaka's complex urban traffic. These limitations also pave the way for future research to refine the system, and address real-world challenges, ultimately improving the applicability of digital twin based traffic management system.

- **Lack of knowledge in urban planning:** Although the study mainly focused on ML-driven vehicle detection and traffic simulation, a deep understanding of urban design and transport policies is required for effective real-world traffic simulation. The lack of specialized knowledge in traffic infrastructure and road capacity planning may have affected attributes such as the accuracy of lane configurations and road priorities in the SUMO simulation. Collaborating with transportation engineers and urban planners in future research would enhance the validity and applicability of the digital twin model for practical urban traffic management.

- **Dataset limitations:** The performance of the YOLO model can be further improved by enriching the dataset with a broader range of vehicle types, ensuring better representation across all categories. Furthermore, images containing null values should be filtered out to enhance data quality, and refining bounding box accuracy will enhance detection precision. To improve the model's adaptability and robustness, it would be beneficial to introduce more variation in image conditions such as lighting, weather, and vehicle occlusions.

- **Lack of computational resources:** Due to computational constraints, we were only able to train the model for 40 epochs. Increasing the number of epochs would likely result in better feature extraction, enhancing the model's overall performance. Additionally, the batch size was restricted to 32, which increased the training time. The underrepresentation of certain vehicle classes in the dataset also hindered the model's ability to detect those objects accurately. Further fine-tuning of hyperparameters could also lead to an increase in model accuracy.

- **Lack of versatile movement detection:** The current system has a limitation in its ability to process vehicles moving horizontally along the frame, as it only accounts for vertical vehicle movement. This creates a challenge when processing video footage of intersections where vehicles are moving in all four directions, as the model cannot properly detect and track the movement. In its current state, four separate recordings of the same intersection are needed to capture all the vehicle movements, which is inefficient. To improve this, implementing a system that accounts for vehicles moving in all directions will significantly enhance data collection for simulation purposes. By processing multi-directional movements in a single recording, we can streamline the data input for the simulation, improving both efficiency and accuracy.

- **Lack of dynamic lane detection:** The current system is limited to the need of a recorded video that has a static viewpoint. The lanes which are placed on the frame are static, as a result a moving viewpoint also moves the lanes, causing data discrepancies and inaccuracy. Implementing dynamic lane data calculation would improve the accuracy of the extracted data and provide flexibility regarding the captured video.

- **No real-time traffic data integration:** The study used YOLOv11 to detect vehicles from pre-recorded videos, limiting the simulation's ability to reflect live traffic conditions dynamically due to the unavailability of real-time traffic data. Without real-time updates, the SUMO and Unity simulations only provide a static representation of traffic at a specific time rather than an evolving, continuously updated model of Dhaka's traffic. A truly real-time adaptive digital twin would require streaming data pipelines for instant decision-making and dynamic route optimizations and direct integration with live CCTV feeds or IoT-based traffic sensors to continuously update SUMO

- **Lack of detailed 3D data:** While Unity was used for 3D visualization, the realism of the digital twin was restricted due to the lack of detailed 3D city models. Instead, the simulation was simplified with plain, simple polygons to represent buildings and structures. As a result, the visual representation lacked the granularity needed for a robust urban simulation. In the future, it is recommended to make a detailed 3D visualization of the traffic networks for a deeper understanding and integration with the Digital Twin.

- **Multidimensional Implementation:** In the future, simulations could be enhanced by integrating Extended Reality (XR) to offer a more immersive and interactive visualization of traffic scenarios. Additionally, the sandbox feature could be made more user-friendly with ongoing development. Creating an all-in-one software solution would greatly support the development of an Intelligent Transportation System (ITS). Improved analytical metrics would provide deeper insights into traffic data, while cloud platforms could be used to manage and host continuous data feeds from live systems. Furthermore, the machine learning model could be advanced to detect license plates and identify vehicles accordingly.

Chapter 8

Conclusion

Urban traffic congestion has long been a significant challenge in Dhaka, Bangladesh. It is one of the most densely populated cities in the world that experiences chronic traffic congestion, inefficient signal management, and inadequate infrastructure. This research has highlighted the pressing challenges of traffic management in Dhaka, Bangladesh, a city with inefficient traffic infrastructure paired with severe congestion. Through intensive research using existing studies and the development of a Digital Twin focused on high-traffic areas such as the Gulshan-2 intersection, we have demonstrated the potential of DT technology integrated with advanced simulation techniques to enhance urban planning and traffic analysis. The project successfully created a dynamic 3D environment with ML-driven simulation mirroring some aspects of real-life traffic scenarios by utilizing tools like Blender, Unity, and SUMO. This paper also highlights the effectiveness of the Digital Twin approach and also shows how the uniqueness of Dhaka city creates newer challenges that needed better understanding.

Our research has demonstrated one of the several possibilities of this implementation. As shown, the study has determined through its visual simulation how the choke points can be identified. It has shown that traffic patterns can be evened out by calibrating waiting times for priority roads. The study also shows a utility method of simulating traffic that can be altered to fit certain problems of transportation. Through small changes in the 3D space and calibration, it can provide insight into the impediment of the real world. It also focuses on how drivers in Bangladesh have an aggressive driving pattern through these visualizations. Simulation in conjunction with real-time data integration can provide valuable insights into traffic flow patterns, peak congestion times, and the impact of various signal times on different intersections, thus laying the foundation for a data-driven approach to urban traffic management in Bangladesh.

Bibliography

[1] R. Rosen, G. von Wichert, G. Lo, and K. D. Bettenhausen, "About the importance of autonomy and digital twins for the future of manufacturing," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 567–572, 2015, 15th IFAC Symposium onInformation Control Problems inManufacturing, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.06.141. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896315003808.

[2] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *arXiv*, 2016. [Online]. Available: https://arxiv.org/abs/1612.08242.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. [Online]. Available: https://arxiv.org/pdf/1506.02640.

[4] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, F.-J. Kahlen, S. Flumerfelt, and A. Alves, Eds. Cham: Springer International Publishing, 2017, pp. 85–113, ISBN: 978-3-319-38756-7. DOI: 10.1007/978-3-319-38756-7_4. [Online]. Available: https://doi.org/10.1007/978-3-319-38756-7_4.

[5] A. .-. A. Kafy, L. Ferdous, S. Poly, *et al.*, "Estimating traffic volume to identify the level of service in major intersections of rajshahi, bangladesh," *Trends in Civil Engineering and its Architecture*, vol. 2, pp. 1–18, 2018. DOI: 10.32474/TCEIA.2018.02.000145.

[6] C. Olaverri-Monreal, J. Errea-Moreno, A. Díaz-Álvarez, C. Biurrun-Quel, L. Serrano-Arriezu, and M. Kuba, "Connection of the sumo microscopic traffic simulator and the unity 3d game engine to evaluate v2x communication-based systems," *Sensors*, vol. 18, no. 12, 2018, ISSN: 1424-8220. DOI: 10.3390/s18124399. [Online]. Available: https://www.mdpi.com/1424-8220/18/12/4399.

[7] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018. [Online]. Available: https://arxiv.org/abs/1804.02767.

[8] M. V., V. V. R., and N. A., "A deep learning rcnn approach for vehicle recognition in traffic surveillance system," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, 2019, pp. 0157–0160. DOI: 10.1109/ICCSP.2019.8698018. [Online]. Available: https://doi.org/10.1109/ICCSP.2019.8698018.

[9] S. Akhter, N. Ahsan, S. J. Quaderi, S. Sumit, M. Forhad, and M. Rahman, "A sumo based simulation framework for intelligent traffic management system," *Transportation Research Part C Emerging Technologies*, vol. 8, pp. 1–5, Jun. 2020. DOI: 10.18178/jtle.8.1.1-5.

[10] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv*, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934.

[11] F. Dembski, U. Wössner, M. Letzgus, M. Ruddat, and C. Yamu, "Urban digital twins for smart cities and citizens: The case study of herrenberg, germany," *Sustainability*, vol. 12, no. 6, p. 2307, 2020. DOI: 10.3390/su12062307.

[12] S. Dorokhin, A. Artemov, D. Likhachev, A. Novikov, and E. Starkov, "Traffic simulation: An analytical review," in *IOP Conference Series: Materials Science and Engineering*, VIII International Scientific Conference Transport of Siberia - 2020, 22-27 May 2020, Novosibirsk, Russia, vol. 918, IOP Publishing Ltd, 2020, p. 012 058. DOI: 10.1088/1757-899X/918/1/012058.

[13] M. R. Islam, M. M. A. Khan, M. M. Hossain, K. K. C. Mani, and R. M. Mat, "Road traffic accidents in bangladesh: Why people have poor knowledge and awareness about traffic rules?" *International Journal of Critical Illness and Injury Science*, vol. 10, no. 2, pp. 70–75, 2020. DOI: 10.4103/IJCIIS.IJCIIS_65_19.

[14] E. Marcucci, V. Gatta, M. Le Pira, L. Hansson, and S. Bråthen, "Digital twins: A critical discussion on their potential for supporting policy-making and planning in urban logistics," *Sustainability*, vol. 12, no. 24, p. 10 623, 2020. DOI: 10.3390/su122410623.

[15] G. Schrotter and C. Hürzeler, "The digital twin of the city of zurich for urban planning," *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 88, no. 1, pp. 99–112, Feb. 2020, ISSN: 2512-2819. DOI: 10.1007/s41064-020-00092-2. [Online]. Available: https://doi.org/10.1007/s41064-020-00092-2.

[16] S. Tabassum, S. Ullah, N. H. Al-nur, and S. Shatabda, "Poribohon-bd: Bangladeshi local vehicle image dataset with annotation for classification," *Data in Brief*, vol. 33, p. 106 465, 2020, ISSN: 2352-3409. DOI: 10.1016/j.dib.2020.106465. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340920313470.

[17] M. Anandhalli, V. Baligar, P. Baligar, and P. Deepsir, "Vehicle detection and tracking for traffic management," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 1, pp. 66–73, 2021. [Online]. Available: https://www.researchgate.net/publication/349731978_Vehicle_detection_and_tracking_for_traffic_management.

[18] L. Bao, Q. Wang, and Y. Jiang, "Review of digital twin for intelligent transportation system," in *2021 International Conference on Information Control, Electrical Engineering and Rail Transit (ICEERT)*, 2021, pp. 309–315. DOI: 10.1109/ICEERT53919.2021.00064.

[19] S. Das, N. Tasnim, M. I. Shihab, M. F. Khan, and R. Ameer, *Team passphrase: Dhaka ai vehicle detection challenge 4th place submission*, Feb. 2021. DOI: 10.13140/RG.2.2.18174.72006.

[20] M. Hämäläinen, "Urban development with dynamic digital twins in helsinki city," *IET Smart Cities*, vol. 3, no. 4, pp. 201–210, 2021. DOI: 10.1049/smc2.12015.

[21] X. Ma, X. Hu, T. Weber, and D. Schramm, "Evaluation of accuracy of traffic flow generation in sumo," *Applied Sciences*, vol. 11, no. 6, 2021, ISSN: 2076-3417. DOI: 10.3390/app11062584. [Online]. Available: https://www.mdpi.com/2076-3417/11/6/2584.

[22] L. Qiu, D. Zhang, and Y. e. a. Tian, "Deep learning-based algorithm for vehicle detection in intelligent transportation systems," *Journal of Supercomputing*, vol. 77, pp. 11 083–11 098, 2021. DOI: 10.1007/s11227-021-03712-9. [Online]. Available: https://doi.org/10.1007/s11227-021-03712-9.

[23] R. Rahman, Z. B. Azad, and M. B. Hasan, "Densely-populated traffic detection using yolov5 and non-maximum suppression ensembling," *arXiv*, Aug. 2021. arXiv: 2108.12118v1 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2108.12118.

[24] A. Rudskoy, I. Ilin, and A. Prokhorov, "Digital twins in the intelligent transport systems," *Transportation Research Procedia*, vol. 54, pp. 927–935, 2021, International Scientific Siberian Transport Forum - TransSiberia 2020, ISSN: 2352-1465. DOI: https://doi.org/10.1016/j.trpro.2021.02.152. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S235214652100332X.

[25] B. D. S. Sai, R. Nikhil, P. S. Babu, and N. S. Naik, "Traffic management using computer vision and sumo," in *2021 IEEE 18th India Council International Conference (INDICON)*, 2021, pp. 1–6. DOI: 10.1109/INDICON52576.2021.9691665.

[26] R. Al-Sehrawy, B. Kumar, and R. Watson, "A digital twin uses classification system for urban planning & city infrastructure management," *Journal of Information Technology in Construction*, vol. 26, pp. 832–862, 2021. DOI: 10.36680/j.itcon.2021.045.

[27] R. M. Alamgir, A. A. Shuvro, M. A. Mushabbir, *et al.*, "Performance analysis of yolo-based architectures for vehicle detection from traffic images in bangladesh," *arXiv*, Dec. 2022. arXiv: 2212.09144v2 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2212.09144.

[28] T. Alghamdi, S. Mostafi, G. Abdelkader, and K. Elgazzar, "A comparative study on traffic modeling techniques for predicting and simulating traffic behavior," *Future Internet*, vol. 14, no. 10, p. 294, 2022, This article belongs to the Special Issue IoT in Intelligent Transportation Systems. DOI: 10.3390/fi14100294.

[29] s. M. S. Bari, R. Islam, and S. Mardia, "Performance evaluation of convolution neural network based object detection model for bangladeshi traffic vehicle detection," in Jan. 2022, pp. 115–128, ISBN: 978-981-16-6635-3. DOI: 10.1007/978-981-16-6636-0_10.

[30] T. Evangelou, M. Gkeli, and C. Potsiou, "Building digital twins for smart cities: A case study in greece," *ISPRS Annals of Photogrammetry Remote Sensing and Spatial Information Sciences*, vol. X-4/W2-2022, pp. 61–68, 2022. DOI: 10.5194/isprs-annals-x-4-w2-2022-61-2022.

[31] M. A. Fattah, S. R. Morshed, and A.-A. Kafy, "Insights into the socio-economic impacts of traffic congestion in the port and industrial areas of chittagong city, bangladesh," *Transportation Engineering*, vol. 9, p. 100 122, 2022, ISSN: 2666-691X. DOI: https://doi.org/10.1016/j.treng.2022.100122. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666691X22000203.

[32] L. Huang and W. Huang, "Rd-yolo: An effective and efficient object detector for roadside perception system," *Sensors*, vol. 22, no. 21, p. 8097, 2022. DOI: 10.3390/s22218097. [Online]. Available: https://doi.org/10.3390/s22218097.

[33] A. Kunz, S. Rosmann, E. Loria, and J. Pirker, "The potential of augmented reality for digital twins: A literature review," in *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2022, pp. 389–398.

[34] C. e. a. Li, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv*, 2022. [Online]. Available: https://arxiv.org/abs/2209.02976.

[35] T. e. a. Liang, "Traffic sign detection via improved sparse r-cnn for autonomous vehicles," *Journal of Advanced Transportation*, vol. 2022, no. 1, p. 3 825 532, 2022. DOI: 10.1155/2022/3825532. [Online]. Available: https://doi.org/10.1155/2022/3825532.

[36] J. Pirker, E. Loria, S. Safikhani, A. Kunz, and S. Rosmann, "Immersive virtual reality for virtual and digital twins: A literature review to identify state of the art and perspectives," in *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, 2022, pp. 114–115.

[37] M. M. Rafi, S. Chakma, A. Mahmud, *et al.*, "Performance analysis of deep learning yolo models for south asian regional vehicle recognition," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 13, no. 9, 2022. DOI: 10.14569/IJACSA.2022.01309100. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2022.01309100.

[38] S. M. Sabbir, "Traffic Volume in Dhaka City at Selected Priority Junctions," *Journal of Transportation Engineering and Traffic Management*, vol. 3, no. 1, Mar. 2022. DOI: 10.5281/zenodo.6330541. [Online]. Available: https://doi.org/10.5281/zenodo.6330541.

[39] U. A. Saima, "Road safety movement: Young adults and collective engagement in social movement," Doctoral dissertation, Brac University, 2022. [Online]. Available: http://hdl.handle.net/10361/18932.

[40] L. Stacchio, A. Angeli, and G. Marfia, "Empowering digital twins with extended reality collaborations," *Virtual Reality & Intelligent Hardware*, vol. 4, no. 6, pp. 487–505, 2022. DOI: 10.1016/j.vrih.2022.06.004.

[41] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv*, 2022. [Online]. Available: https://arxiv.org/abs/2207.02696.

[42] J. Wu, X. Wang, Y. Dang, and Z. Lv, "Digital twins and artificial intelligence in transportation infrastructure: Classification, application, and future research directions," *Computers and Electrical Engineering*, vol. 101, p. 107 983, 2022, ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2022.107983. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790622002488.

[43] Y. Ali, M. Rafay, R. D. A. Khan, M. K. Sorn, and H. Jiang, "Traffic problems in dhaka city: Causes, effects, and solutions (case study to develop a business model)," *Open Access Library Journal*, vol. 10, no. 5, 2023. DOI: 10.4236/oalib.1109994.

[44] R. Chai, Y. Hou, B. Fu, and Y. He, "Simulation modeling of typical urban traffic congestion areas based on sumo," in *2023 6th International Conference on Information Communication and Signal Processing (ICICSP)*, 2023, pp. 853–859. DOI: 10.1109/ICICSP59554.2023.10390622.

[45] S. Dasgupta, M. Rahman, A. D. Lidbe, W. Lu, and S. Jones, *A transportation digital-twin approach for adaptive traffic control systems*, 2023. arXiv: 2109.10863 [cs.SY]. [Online]. Available: https://arxiv.org/abs/2109.10863.

[46] D. A. Guastella and G. Bontempi, "Traffic modeling with sumo: A tutorial," *arXiv*, Mar. 2023. arXiv: 2304.05982v1 [cs.NI]. [Online]. Available: https://arxiv.org/abs/2304.05982.

[47] M. A. Hossain, M. A. Rahman, R. Khanom, S. Sultana, and A. Hossain, "Real-time vehicle detection and classification on the padma multipurpose bridge in bangladesh using a deep learning model," in *2023 International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM)*, 2023, pp. 1–6. DOI: 10.1109/NCIM59001.2023.10212549.

[48] Z. Huang, L. Li, G. C. Krizek, and L. Sun, "Research on traffic sign detection based on improved yolov8," *Journal of Computer and Communications*, vol. 11, no. 7, 2023. [Online]. Available: https://www.scirp.org/reference/referencespapers?referenceid=3532980.

[49] M. Jafari, A. Kavousi-Fard, T. Chen, and M. Karimi, "A review on digital twin technology in smart grid, transportation system and smart city: Challenges and future," *IEEE Access*, vol. 11, pp. 17 471–17 484, 2023. DOI: 10.1109/ACCESS.2023.3241588.

[50] B. Jang, I. Yoo, and D. Yook, "Pipelined stochastic gradient descent with taylor expansion," *Applied Sciences*, vol. 13, no. 21, p. 11 730, 2023. DOI: 10.3390/app132111730.

[51] V. K. Kumarasamy, A. J. Saroj, Y. Liang, *et al.*, "Traffic signal optimization by integrating reinforcement learning and digital twins," in *2023 IEEE Smart World Congress (SWC)*, 2023, pp. 1–8. DOI: 10.1109/SWC57546.2023.10448974.

[52] G. Lavanya and S. Pande, "Enhancing real-time object detection with yolo algorithm," *EAI Endorsed Transactions on Internet of Things*, vol. 10, 2023. [Online]. Available: https://www.researchgate.net/publication/376252973_Enhancing_Real-time_Object_Detection_with_YOLO_Algorithm.

[53] N. S. Munir, N. Hossain, R. R. Zame, and G. Sarowar, "Vehicle detection of bangladesh using yolov7 with hyper-parameter tuning," *2023 3rd International conference on Artificial Intelligence and Signal Processing (AISP)*, pp. 1–5, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259028203.

[54] Z. Rezaee, M. H. Vahidnia, H. Aghamohammadi, Z. Azizi, and S. Behzadi, "Digital twins and 3d information modeling in a smart city for traffic controlling: A review," *Journal of Geography and Cartography*, vol. 6, pp. 1–27, Jun. 2023. DOI: 10.24294/jgc.v6i1.1865.

[55] M. S. I. Sakib, "Bangladesh and it's traffic congestion," Jan. 2023.

[56] I. Sarker, S. Rahman, N. K. Nuha, *et al.*, "Bangladeshi vehicle classification using transfer learning with yolov7," in *2023 6th International Conference on Electrical Information and Communication Technology (EICT)*, 2023, pp. 1–6. DOI: 10.1109/EICT61409.2023.10427682.

[57] N. Slob, W. Hurst, R. van de Zedde, and B. Tekinerdogan, "Virtual reality-based digital twins for greenhouses: A focus on human interaction," *Computers and Electronics in Agriculture*, vol. 208, p. 107 815, 2023. DOI: 10.1016/j.compag.2023.107815.

[58] H. Xu, A. Berres, S. B. Yoginath, *et al.*, "Smart mobility in the cloud: Enabling real-time situational awareness and cyber-physical control through a digital twin for traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 3, pp. 3145–3156, 2023. DOI: 10.1109/TITS.2022.3226746.

[59] H. Yeon, T. Eom, K. Jang, *et al.*, "Dtumos, digital twin for large-scale urban mobility operating system," *Scientific Reports*, vol. 13, p. 5154, 2023. DOI: 10.1038/s41598-023-32326-9.

[60] R. F. El-Agamy, H. A. Sayed, A. M. AL Akhatatneh, M. Aljohani, and M. Elhosseini, "Comprehensive analysis of digital twins in smart cities: A 4200-paper bibliometric study," *Artificial Intelligence Review*, vol. 57, no. 6, 2024. DOI: 10.1007/s10462-024-10781-8.

[61] M. A. R. Alif, "Yolov11 for vehicle detection: Advancements, performance, and applications in intelligent transportation systems," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2410.22898.

[62] E. Aloupogianni, F. Doctor, C. Karyotis, T. Maniak, R. Tang, and R. Iqbal, "An ai-based digital twin framework for intelligent traffic management in singapore," in *2024 International Conference on Electrical, Computer and Energy Technologies (ICECET*, 2024, pp. 1–6. DOI: 10.1109/ICECET61485.2024.10698642.

[63] S. Dasgupta, M. Rahman, and S. Jones, "Harnessing digital twin technology for adaptive traffic signal control: Improving signalized intersection performance and user satisfaction," *IEEE Internet of Things Journal*, pp. 1–1, 2024. DOI: 10.1109/JIOT.2024.3420439.

[64] O. Dobrilko and A. Bublil, *Leveraging sumo for real-world traffic optimization: A comprehensive approach*, SUMO Documentation, Apr. 2024. [Online]. Available: https://sumo.dlr.de/pdf/2024/1-2.pdf.

[65] B. Dwyer, J. Nelson, T. Hansen, *et al.*, *Roboflow (version 1.0)*, Computer vision software, 2024. [Online]. Available: https://roboflow.com.

[66] S. A. Fahim, "Finetuning yolov9 for vehicle detection: Deep learning for intelligent transportation systems in dhaka, bangladesh," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2410.08230.

[67] M. M. Faisal, *Yolov8 tracking, counting, and speed estimation*, GitHub repository, 2024. [Online]. Available: https://github.com/MuhammadMoinFaisal/Computervisionprojects/blob/main/YOLOv8_Tracking_Counting_SpeedEstimation/predict.py.

[68] A. Ghosh, *Yolo11: Faster than you can imagine!* LearnOpenCV, Oct. 2024. [Online]. Available: https://learnopencv.com/yolo11/.

[69] N. Jegham, C. Y. Koh, M. Abdelatti, and A. Hendawi, "Evaluating the evolution of yolo (you only look once) models: A comprehensive benchmark study of yolo11 and its predecessors," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2411.00201.

[70] S. Khadka, P. (Wang, P. (Li, and S. P. Mattingly, "Automated traffic signal performance measures (atspms) in the loop simulation: A digital twin approach," *Transportation Research Record*, vol. 0, no. 0, 2024. DOI: 10.1177/03611981241258985.

[71] R. Khanam and M. Hussain, "What is yolov5: A deep look into the internal features of the popular object detector," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2407.20892.

[72] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2410.17725.

[73] R. Khanam, M. Hussain, R. Hill, and P. Allen, "A comprehensive review of convolutional neural networks for defect detection in industrial applications," *IEEE Access*, 2024. DOI: 10.1109/ACCESS.2024.3425166. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3425166.

[74] A. Mohammadi, P. Y. Park, M. Nourinejad, M. S. B. Cherakkatil, and H. S. Park, "Sumo2unity: An open-source traffic co-simulation tool to improve road safety," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, 2024, pp. 2523–2528. DOI: 10.1109/IV55156.2024.10588571.

[75] S. Mupparaju, R. Thotakura, and C. Venkata, "A review on yolov8 and its advancements," in *Data Intelligence and Cognitive Informatics*, 2024, pp. 529–545. DOI: 10.1007/978-981-99-7962-2_39. [Online]. Available: https://doi.org/10.1007/978-981-99-7962-2_39.

[76] S. Patil, S. Waghule, S. Waje, P. Pawar, and S. Domb, "Efficient object detection with yolo: A comprehensive guide," *International Journal of Advanced Research in Science, Communication and Technology*, pp. 519–531, 2024. [Online]. Available: https://www.researchgate.net/publication/380876424_Efficient_Object_Detection_with_YOLO_A_Comprehensive_Guide.

[77] M. Pechinger and J. Lindner, "Sumonity: Bridging sumo and unity for enhanced traffic simulation experiences," *SUMO Conference Proceedings*, vol. 5, pp. 163–177, Jul. 2024. DOI: 10.52825/scp.v5i.1115.

[78] S. U. Salekin, M. H. Ullah, A. A. A. Khan, M. S. Jalal, H.-H. Nguyen, and D. M. Farid, "Bangladeshi native vehicle classification employing yolov8," in *Intelligent Systems and Data Science*, ser. Communications in Computer and Information Science, N. Thai-Nghe, T.-N. Do, and P. Haddawy, Eds., vol. 1949, Springer, Singapore, 2024. DOI: 10.1007/978-981-99-7649-2_14. [Online]. Available: https://doi.org/10.1007/978-981-99-7649-2_14.

[79] M. R. Sarkar, M. A. Hossain, and S. M. S. Ahamed, "Possible causes and solutions of the traffic jam in dhaka," *European Journal of Social Sciences Studies*, vol. 9, no. 6, p. 74, Apr. 2024. DOI: 10.46827/ejsss.v9i6.1690.

[80] Ultralytics, *Ultralytics yolov11*, Online documentation, Accessed: 21-Oct-2024, 2024. [Online]. Available: https://docs.ultralytics.com/models/yolo11/.

[81] C.-Y. Wang and H.-Y. M. Liao, "Yolov1 to yolov10: The fastest and most accurate real-time object detection systems," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2408.09332.

[82] P. Hidayatullah, N. Syakrani, M. R. Sholahuddin, T. Gelar, and R. Tubagus, "Yolov8 to yolo11: A comprehensive architecture in-depth comparative review," *arXiv*, 2025. DOI: 10.48550/arXiv.2501.13400. [Online]. Available: https://doi.org/10.48550/arXiv.2501.13400.