Tribhuvan University

Institute of Science and Technology

A Final Year Project Report

On

Autonomous Driving Vehicle [CSC412]

Submitted To

Department of Information Technology

Kathford International College of Engineering and Management

In partial fulfillment of the requirement for the Bachelor Degree in Computer Science and Information Technology

Submitted By

Abhijeet Pandey (TU Exam Roll No. 23373/076)

Amrit Kharel (TU Exam Roll No. 23377/076)

Bishal Pahari (TU Exam Roll No. 23386/076)

Under the Supervision of

Mr. Ishwar Dhungana

1st March, 2024

ACKNOWLEDGEMENT

We express our deep gratitude to our esteemed supervisor, Mr. Ishwar Dhungana, whose unwavering guidance has played a pivotal role in shaping the trajectory of our final year project on the implementation of an ADV in GTA V. His perceptive feedback, constructive critique, and unwavering support have played a pivotal role in pushing our project forward. Also, we would like to express our sincere appreciation to our friend, Saurav Bhattarai, for his valuable contribution. His dedication and collaborative spirit have significantly enhanced the depth and quality of our project. We would also like to acknowledge the valuable contributions of Harrison Kinsley and Rudi Grobler, who have provided technical assistance and insights that have proven to be indispensable in overcoming challenges and refining our implementation. The collective efforts of our team members have fostered a collaborative synergy that is crucial for navigating the intricate landscape of developing an ADV within the unique context of GTA V. As we approach the mid-defense stage, we have full confidence that the mentorship and contributions from these individuals will continue to guide us toward a successful culmination of our project, setting the stage for its continued advancement in subsequent phases.

ABSTRACT

In a constantly changing and evolving global landscape where the prevalence of AI is widespread, the concept of autonomous cars seems to be within reach. The ADV, currently being developed as a driver-assistance system, showcases a fascinating self-driving ability. This technology presently encompasses the ability to maintain its position within a lane and adjust its speed in response to traffic conditions, without necessitating human intervention. The implementation of the ADV takes place within a simulated environment, specifically the virtual realm known as GTA V. Within this virtual domain, an accurate representation of the real-world environment is meticulously recreated, complete with traffic signals, motor vehicles, public transport, and road signs, among other elements. The ultimate objective of the ADV is to navigate this simulated environment, to attain an autonomous operation.

Keywords: ADV, AI, Lane Maintenance, Speed adjustments, Simulated environment

Table of Contents

No table of figures entries found.

List of Abbreviations

ADV	Autonomous Driving Vehicle
AI	Artificial Intelligence
CNN	Convolution Neural Network
GIS	Geographical Information System
GTA	Grand Theft Auto
ML	Machine Learning
ReLU	Rectified Linear Unit
TDD	Test Driven Development
XP	Extreme Programming

1. Introduction

1.1 Introduction

In today's changing world of car technology, making cars that can drive themselves has become a transformative idea that can change how we travel. This report explores the idea of having ADVs in GTA V. When we think about it, it's where smart computer technology and the big dreams of car companies come together.

Right now, we're seeing many advancements in smart technology, especially in AI. ADVs aren't just a concept anymore, they're becoming a real thing that can make driving safer and easier for everyone. Car companies are competing to make cars that can drive themselves, freeing people from boring tasks and making travel better for all. It's like a future where car crashes are rare, traffic isn't a problem, and anyone can go anywhere easily. It's a future where computers help us drive, giving us more freedom.

Now, let's talk about integrating ADVs in GTA V. GTA V is all about having fun and providing an immersive open-world dynamic environment. It’s a great place where computers can learn how to drive a car in a real, dynamic environment. Within this game environment, we can train computer programs to drive in tough situations, just like in real life. This report will look at how we can try to achieve it. In these virtual environments, we have attempted to simulate ADV capabilities.

1.2 Problem Statement

In real life, one of the pressing issues in transportation is the need to improve road safety and reduce accidents. Despite advancements in technology, human error remains a significant factor contributing to road accidents. Additionally, traffic congestion continues to be a major concern in urban areas, leading to inefficiencies in transportation systems and increased travel times for commuters.

How Level 2 Autonomy Could Help:

Partial level 2 autonomy, such as the system currently implemented in GTA 5, could offer solutions to some of these real-life challenges. By enabling vehicles to autonomously handle certain driving tasks, such as maintaining lane position and following a predetermined route, partial level 2 autonomy could potentially reduce the likelihood of accidents caused by human error. Moreover, it could contribute to smoother traffic flow and alleviate congestion by optimizing driving behavior in urban environments.

How Our Project Could Contribute:

While the current implementation in GTA 5 lacks certain capabilities such as consistent adherence to traffic rules and object detection, our project aims to address these deficiencies. By focusing on improving the behavior of ADV within the game, we seek to enhance their ability to follow traffic rules, detect and avoid obstacles, and prioritize safety during navigation. Through research and development efforts, we aim to advance the capabilities of Autonomous Driving technology within the gaming environment, laying the groundwork for potential real-world applications. By addressing these limitations, our project contributes to the broader goal of improving road safety and efficiency in transportation systems, both virtually in GTA 5 and potentially in real-life scenarios.

1.3 Objectives

1. To make an autonomous driving car in a virtual environment (GTA V).
2. To develop a mobile interface that can be used to communicate with an autonomous car.

1.4 Scope and Limitations

1.4.1 Scope

Our ADV project in GTA V involves designing an intuitive system within the game for autonomous driving. This includes developing an easy-to-use interface for users to control

and manage the ADV efficiently. The aim is to implement Autonomous driving in a virtual environment by allowing users to update destinations, and other relevant details seamlessly. The Major scopes of our project are:

- **GPS Path Follow:** Implement a system where autonomous vehicles in GTA follow predefined GPS paths and navigate throughout the game world.
- **Real-time Decision Making:** Develop algorithms capable of making real-time decisions based on dynamic changes in the game environment, such as traffic conditions, pedestrian movement, and unexpected obstacles.
- **Optimized for Low-Powered Standard Vehicles (Non-Sports):** Design the autonomous vehicle system to operate efficiently with low-powered standard vehicles commonly found in the game, ensuring optimal performance without relying on high-speed or sports car attributes.
- **Forward Driving Only**: Restrict the autonomous vehicles to operate solely in forward driving mode, aligning with typical traffic regulations and ensuring simplicity in navigation algorithms.
- **Optimized for Clear Weather and Daylight:** Enhance the system's performance to function optimally under clear weather conditions and during daylight hours, minimizing potential disruptions due to adverse weather or low visibility.
- **Priorly Optimized Well-Marked Roads:** Focus on roads that are well-marked within the game environment, prioritizing navigation and decision-making on these routes to ensure smoother and more reliable autonomous vehicle operation.

1.4.2 Limitations

Different limitations of this project are:

- The vehicle does not consistently comply with traffic rules, which can limit the realism and safety of the autonomous driving experience.
- The AI system is not very effective at avoiding collisions, indicating a need for improvement in its object detection capabilities.
- The vehicle can exhibit unexpected behavior during operation, disrupting system effectiveness.
- Unrealistic AI decisions can result from sudden environmental changes.
- Gathering accurate training data in a virtual environment is tough, differing from real-world scenarios.

- While the project aims to implement the self-driving experience within GTA V, expanding the project to cover more scenarios or environments may need extra resources.
- Continuous support and updates are crucial for the sustained success of the virtual autonomous driving system in GTA V.
- ADV cannot perfectly navigate complex city junctions, potentially affecting system performance.
- Inability to operate effectively in nighttime conditions due to reduced visibility and sensor limitations.

1.5 Development Methodology

Extreme Programming (XP) methodology is applied in the development process of our Autonomous Driving project. We chose XP for our project because of its emphasis on collaboration, flexibility, and iterative development. As a team of three members, we recognized the importance of working closely together and adapting to changes quickly. XP's practices, such as daily stand-up meetings and continuous planning, allowed us to stay aligned on our goals and make progress efficiently.

The fundamental principles of Extreme Programming, including incremental planning, frequent testing, continuous integration, and shared ownership, serve as guiding pillars for our development process. These principles foster a dynamic and adaptive environment for our project team.

1.5.1 Incremental Planning

We break down our project into smaller, manageable tasks and plan our development incrementally. This approach allows us to focus on achievable goals within shorter time frames, ensuring steady progress towards our ultimate objective of developing an autonomous driving system within GTA 5.

1.5.2 Frequent Testing

We conduct tests regularly to verify the functionality and performance of our autonomous driving system. By testing frequently, we can identify and address issues early in the development cycle, minimizing the risk of larger problems arising later on.

1.5.3 Continuous Integration

We integrate new code changes into our project repository frequently, ensuring that all team members are working with the most up-to-date version of the software. This practice promotes collaboration and helps us identify and resolve integration issues promptly.

1.5.4 Shared Ownership

Every member of our project team takes ownership of the codebase and shares responsibility for its success. This collaborative approach encourages communication, knowledge sharing, and collective problem-solving, ultimately leading to a stronger and more resilient development team.

1.6 Report Organization

The Report has been prepared following the guidelines provided by Tribhuvan University. The report is separated into different chapters. Each chapter consists of various sub-chapters with its content. The preliminary section of the report consists of a Title Page, Acknowledgement, Abstract, Table of Contents, List of Abbreviations, List of Figures, and List of tables.

The main report is divided into 6 chapters, which include:

Certainly, here's a brief explanation for each main topic based on the given structure:

1. Chapter 1: Introduction

This chapter sets the stage for the entire project, providing a general overview, the motivation behind the project, and the problem it aims to solve. It outlines the specific objectives the project seeks to achieve, clarifies the scope and limitations to manage expectations, and describes the development methodology that guides the project process. Finally, it presents the organization of the report, detailing what each subsequent chapter will cover.

2. Chapter 2: Background Study and Literature Review

This section delves into the theoretical foundations and terminologies pertinent to the project, ensuring the reader has a clear understanding of the project's technical context. It includes a comprehensive literature review that evaluates similar projects, theories, and

findings from other researchers, highlighting how this project aligns with or diverges from existing work and identifying gaps the project intends to fill.

3. Chapter 3: System Analysis

This section conducts a thorough analysis of the system, starting with requirement analysis that divides into functional and non-functional requirements, providing clarity on what the system is expected to do and how it should perform. The feasibility analysis assesses the project's viability in terms of technical, operational, economic, and schedule perspectives. Depending on the chosen approach (structured or object-oriented), this chapter outlines the initial design models that serve as a blueprint for the system's development.

4. Chapter 4: System Design

The focus of this chapter is on translating the requirements and analysis from the previous chapter into detailed design specifications. Depending on the analysis approach, it either details the database, forms, reports, and interface designs or refines the object-oriented models. Additionally, it discusses algorithm details essential for the implementation phase, ensuring that the system's architecture is robust and scalable.

5. Chapter 5: Implementation and Testing

This chapter documents the transition from design to actual implementation, listing the tools, programming languages, and databases used in the project. It provides detailed descriptions of the implementation of various modules, highlighting significant classes, methods, and algorithms. The chapter also covers exhaustive testing procedures, including unit and system testing, to ensure reliability and functionality, followed by an analysis of the results.

6. Chapter 6: Conclusion and Future Recommendations

The final chapter wraps up the project with a summary of findings, achievements, and the extent to which the project objectives were met. It reflects on the project's contribution to the field and discusses any limitations encountered. The chapter concludes with recommendations for future work, suggesting areas for further research or improvement to build upon the project's achievements.

2. Background Study and Literature Review

2.1 Background Study

In the world of autonomous driving, many existing methods struggle with outdated technology, causing delays and a lack of up-to-date information. This can make the experience for passengers less enjoyable. To tackle these issues, modern autonomous driving systems use advanced sensors, artificial intelligence, and real-time data analysis to make transportation safer and more efficient.

In our ADV project, we've noticed similar limitations in existing systems, leading to dissatisfaction among users and inefficiencies in operations. Our goal is to create a better autonomous driving system that overcomes these problems. Our system will give real-time updates to destination details through mobile apps.

To make our system even better, we'll use machine learning techniques. These techniques will look at training data and try to create a better-performing ADV. By giving useful information, our system hopes to make autonomous driving smoother and more efficient.

In conclusion, our project aims to fix current problems by making an improved autonomous driving system. It'll be user-friendly, and give real-time updates. We understand that our project is part of academic research and see the potential for making small improvements in autonomous vehicle technology over time.

2.2 Literature Review

1. **Waymo:** Waymo is part of Alphabet Inc. (used to be known as Google's ADV project), and has been right at the front of fancy ADV tech. They've spent lots of time doing research and making cool stuff. Waymo is a big name in the business of cars that drive themselves. They've worked hard to make sure their ADVs are safe and smart enough to handle tricky city roads. In 2018, they did something big - they started a taxi service called Waymo One with ADVs. This was a big deal because it showed how ADVs can be used in our daily lives. Waymo's been testing their cars a lot, and that's taught us a bunch about what's good and tough about self-

driving. As Waymo keeps improving and making more ADVs, they're showing us how the future of getting around might look for all of us [1].

2. **Python plays GTA:** In 2018, the founder of pythonprogramming.net, Harrison Kinsley, also known as Sentdex, endeavored to achieve a self-autonomous car in the GTA V game using Python. He utilized multiple techniques to realize an ADV, commencing with the foundational step of creating a functional interface. Progressing through successive stages, Kinsley incorporated advanced concepts, including convolutional neural networks (CNNs). Documented in his "Python Plays GTA" series, his journey not only showcases his technical prowess but also serves as a comprehensive exploration of the intersection between programming and AI. Through iterative experimentation, he improved his model, introducing version 2 of his car, called "Charles 2.0." In this iteration, he implemented ALEXNET to demonstrate how enhanced neural network architectures contribute to the autonomous capabilities of the vehicle. This initiative stands as a testament to Kinsley's innovative approach, pushing the boundaries of AI applications within the dynamic context of a popular gaming platform [2].

3. **Beyond GTA V for Training, Testing, and Enhancing Deep Learning in Self-Driving Cars by Princeton University:** In 2018, Princeton University initiated a groundbreaking exploration into the utilization of virtual environments, particularly GTA V (GTA-V), for the training, testing, and enhancement of deep learning in ADVs. The research generated a dataset of over 480,000 labeled virtual images depicting normal highway driving, using CNNs and RNNs to train models for essential autonomous driving variables. Testing on 50,000 images from diverse GTA-V driving environments yielded encouraging results. This initial assessment highlights the efficiency and flexibility of a "GTA-V"-like virtual environment, offering a well-defined foundation for training and testing Convolutional Neural Networks and Recurrent Neural Networks in the pursuit of safe autonomous driving [3].

3. System Analysis

3.1 Requirement Analysis

3.1.1 Functional Requirements

1) **Vehicle Control:**

   Ensure that the ADV can proficiently perform fundamental vehicle control tasks, including acceleration, deceleration, and steering within the virtual environment. It should allow the user to override vehicle controls at any point.

2) **Obstacle Detection and Avoidance:**

   Utilize a camera to consistently scan the virtual environment for obstacles. Demonstrate adequate capability to make real-time decisions and take action to prevent collisions.

3) **User Interface:**

   Develop an intuitive Graphical User Interface (GUI) that allows users to set destinations and monitor the ADV's movement. The GUI displays information such as the vehicle's current location, destination marker, available location, and should enable it to track the vehicle's current location in real time.

Following is the Use Case diagram for Autonomous driving vehicle:

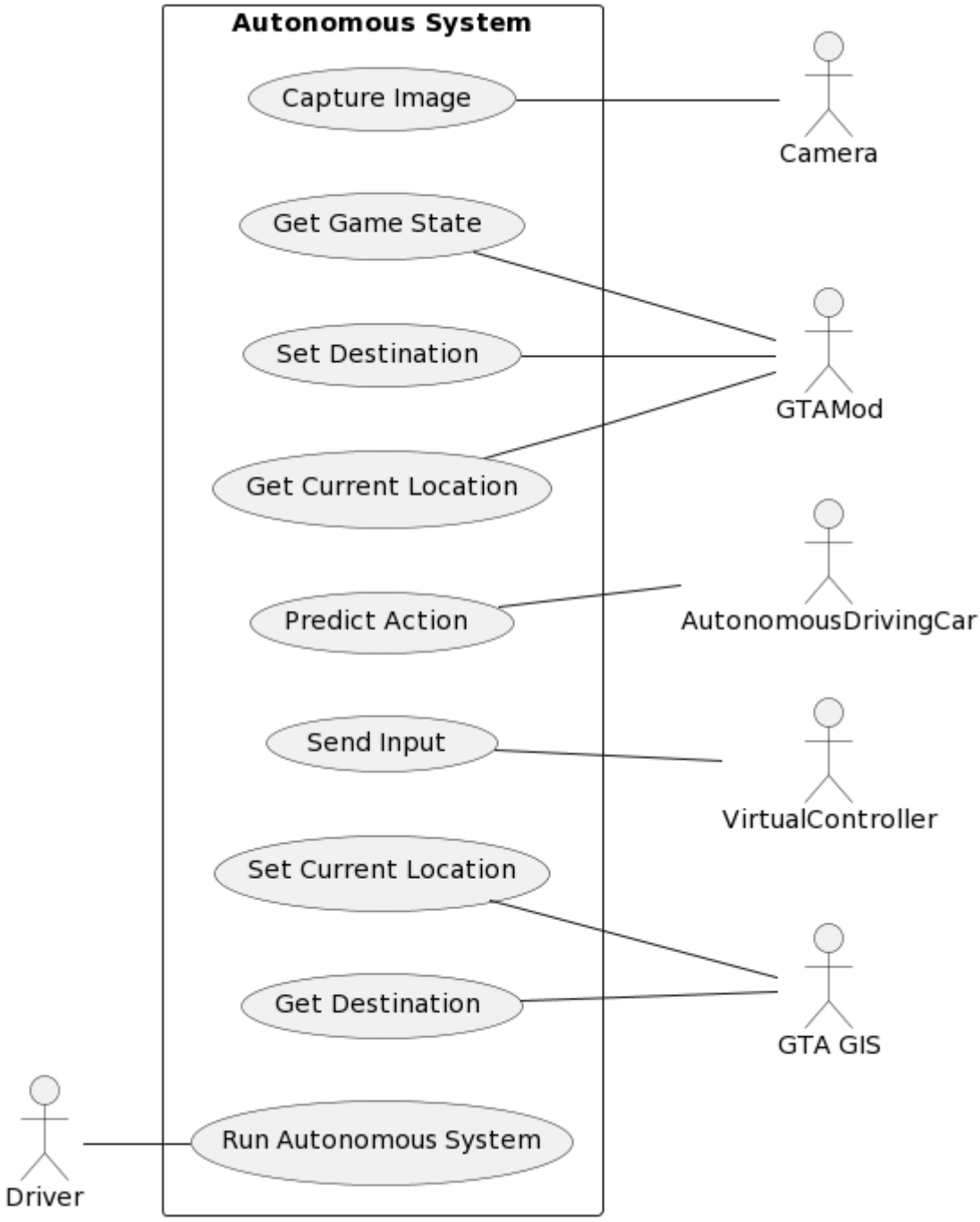


Figure 1: Use Case Diagram of ADV

3.1.2 Non-Functional Requirements

1) Performance:

Ensure the ADV system maintains responsive performance with low-latency interactions in the virtual environment.

2) Scalability:

Design the system to effectively manage a growing number of virtual vehicles, pedestrians, and intricate traffic scenarios without experiencing significant performance degradation.

3) Realism:

Replicate real-world driving conditions within the virtual environment.

4) Usability:

Create a user-friendly interface with intuitive controls and informative feedback. Users should be able to quickly comprehend and interact with the ADV and simulation scenarios.

3.2 Feasibility Analysis

3.2.1 Technical Feasibility:

Creating an ADV in GTA V using software resources is feasible with ML and computer vision. However, road segmentation and object detection pose challenges, requiring considerable processing time and computationally capable hardware.

3.2.2 Operational Feasibility:

The ADV in GTA V uses software to control in-game vehicles, analyzing data like vehicle positions and road conditions. ML algorithms will then make driving decisions based on this data. For ease of use, there'll be a user-friendly interface for selecting destinations.

3.2.3 Economic Feasibility:

The economic feasibility of our project is primarily dependent on the purchase of the GTA game, which serves as the main expense. By utilizing existing software resources like ML

and computer vision, we minimize additional costs, making our project economically viable.

3.2.4 Schedule Feasibility:

The Autonomous Vehicle project has been successfully completed and concluded according to the predetermined timeline.

4. System Design

4.1 Design

4.1.1 Refinement of Class, Object, State, Sequence, and Activity Diagrams

1) Class Diagram Analysis:

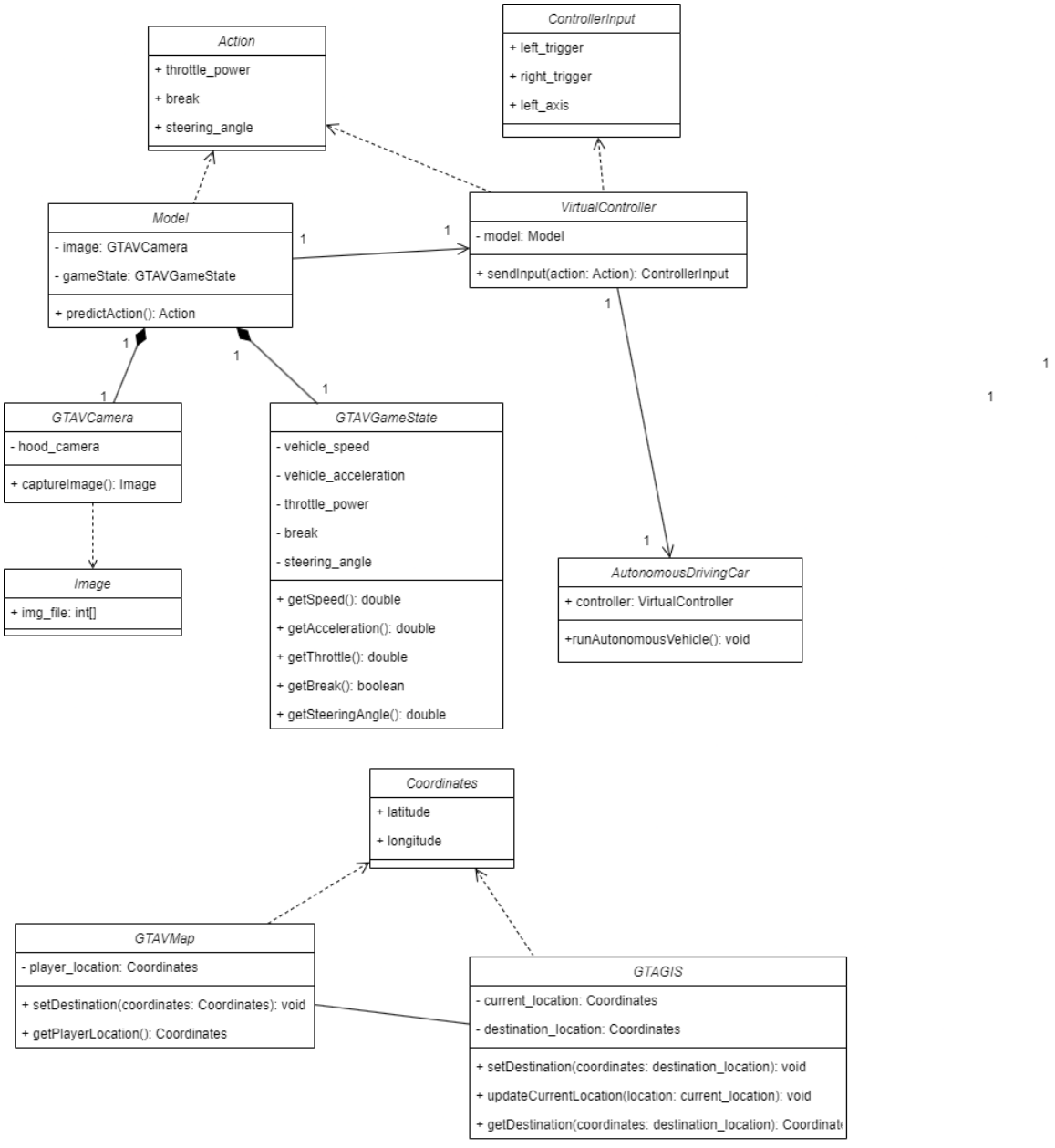


Figure 2: Class Diagram of ADV

This class diagram represents the components of an Autonomous Driving Car system designed for the GTA V (GTA V) environment.

GTAVCamera:

- Manages front and back cameras for capturing images.
- Provides a method `captureImage()` to capture images.

GTAVGameState:

- Stores information about the game state, including vehicle speed, acceleration, throttle power, brake, and steering angle.
- Has a method `getGameState()` to retrieve the current game state.

GTAVMap:

- Handles player location on the map.
- Includes methods to set a destination using coordinates and retrieve the player's location.

Model:

- Uses captured images and game state information to predict the next action.
- Has a method `predictAction()` that takes an image and game state, returning the predicted action.

VirtualController:

- Sends input commands to control the vehicle in the virtual environment.
- Provides a method `sendInput()` to send actions to control the car.

GTAGIS (Geographic Information System for GTA):

- Manages the current and destination locations within the GTA environment.
- Includes methods to set and get destination locations and set the current location.

AutonomousDrivingCar:

- Integrates all components (GTAVCamera, GTAVGameState, Model, VirtualController, GTAGIS) to run the autonomous driving system.
- Has a method `runAutonomousSystem()` to execute the autonomous driving functions.

Coordinates:

- Represents the x and y coordinates on the map.

Action:

- Defines actions such as throttle power, brake, and steering angle.

Arrows indicate relationships between classes, illustrating how they interact within the autonomous driving system in GTA V.

2) Object Diagram Analysis

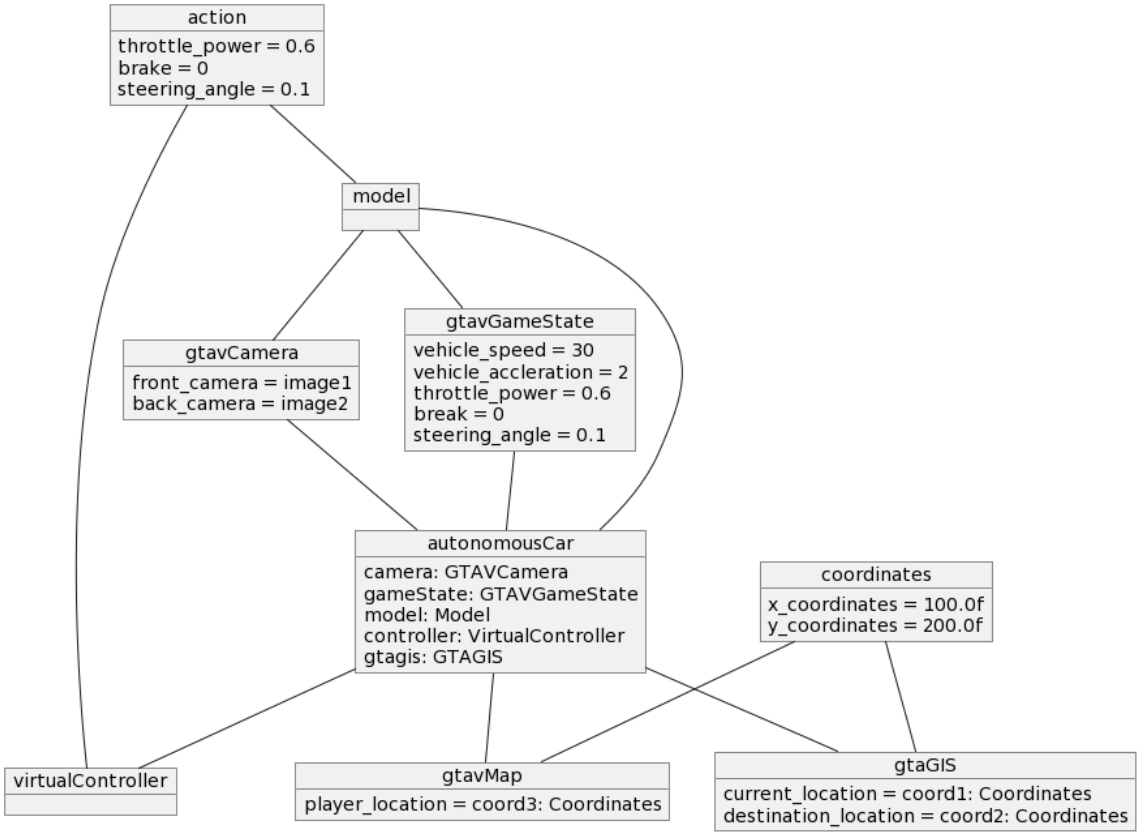


Figure 3: Object Diagram of ADV

This object diagram represents the various objects and their relationships within an ADV system designed in the GTA V (GTA V) environment.

autonomousCar:

- Represents the overarching object for the Autonomous Driving Car system.
- Comprises components such as camera, gameState, model, controller, gta-gis, and their interconnections.

gtavCamera:

- Represents the GTAVCamera object with front_camera and back_camera attributes storing images.

gtavGameState:

- Represents the GTAVGameState object with attributes like vehicle_speed, vehicle_acceleration, throttle_power, brake, and steering_angle.

Model:

- Represents the Model object responsible for predicting actions.

virtualController:

- Represents the VirtualController object handling input commands.

gtaGIS:

- Represents the GTAGIS object managing current and destination locations using coordinates.

Coordinates:

- Represents the Coordinates object with x_coordinates and y_coordinates attributes.

Action:

- Represents the Action object with attributes for throttle_power, brake, and steering_angle.

gtavMap:

- Represents the GTAVMap object with player_location attribute storing coordinates.

Relationships:

- The autonomousCar object connects to various components, including virtualController, gtaGIS, and gtavMap.
- The gtavCamera, gtavGameState, model, virtualController, and gtaGIS objects are interconnected with the autonomousCar object.
- The model connects to gtavCamera and gtavGameState.
- Coordinates are used by gtaGIS and gtavMap.
- Actions are utilized by both virtualController and model.

This object diagram provides a visual overview of the system's structure and the relationships between different components in the Autonomous Driving Car system within the GTA V environment.

3) State Diagram Analysis:

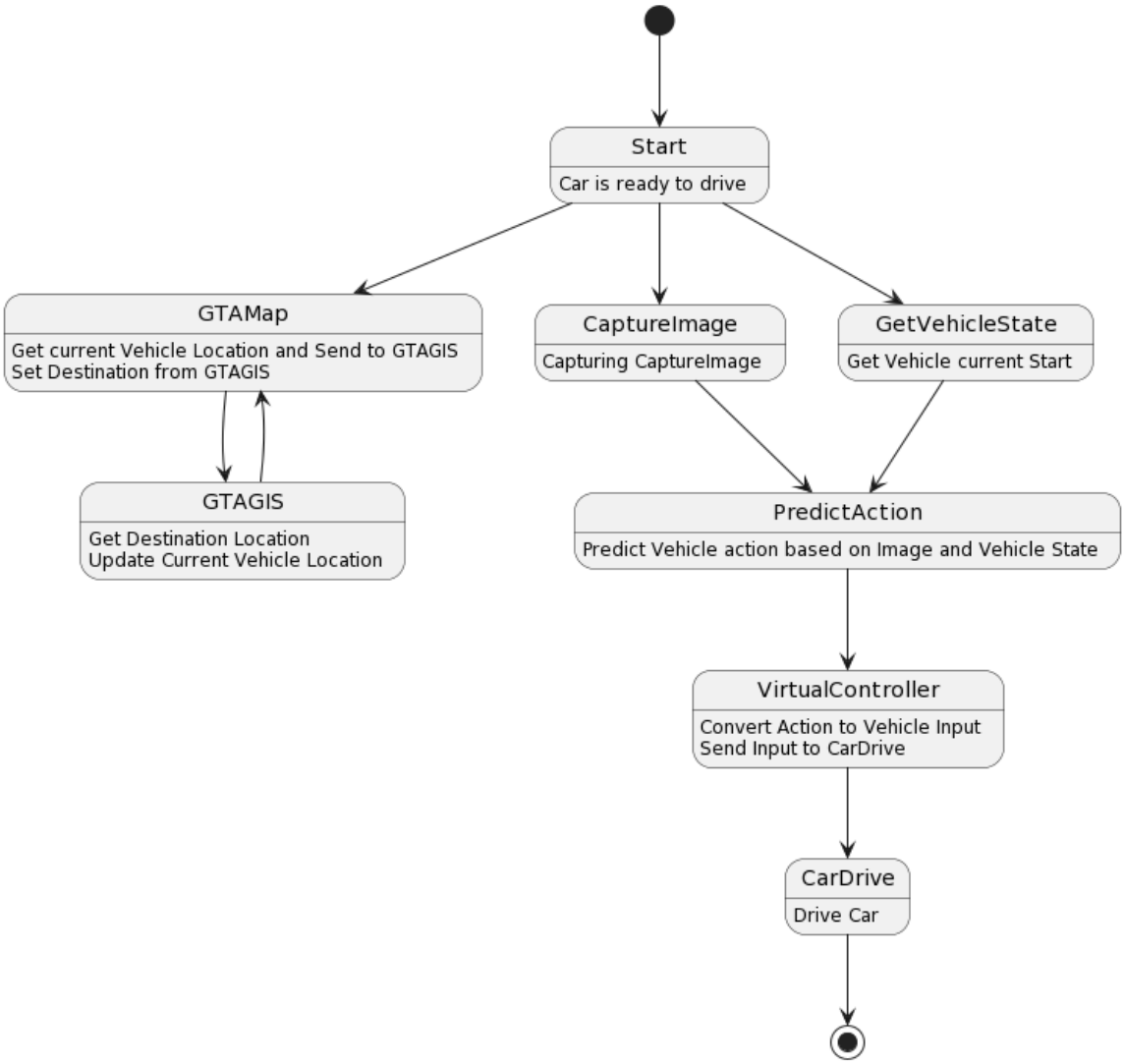


Figure 4: State Diagram of ADV

This state diagram outlines the sequence of states and transitions in a system representing the process of preparing a car for autonomous driving and executing actions within the GTA (GTA) environment.

Start State:

- Represents the initial state where the car is ready to drive.

- Transition: Leads to the GTAMap state.

GTAMap State:

- Represents the state where the system interacts with the GTAMap module.
- Actions:
    - Gets the current vehicle location and sends it to GTAGIS.
    - Sets the destination based on information from GTAGIS.
- Transitions:
    - Leads to interactions with GTAGIS.

GTAGIS State:

- Represents the state where the system interacts with the GTAGIS module.
- Actions:
    - Gets the destination location.
    - Updates the current vehicle location.
- Transitions:
    - Leads back to GTAMap, forming a continuous loop.

CaptureImage State:

- Represents the state of capturing images.
- Action:
    - Captures images for further processing.
- Transition:
    - Leads to the PredictAction state.

GetVehicleState State:

- Represents the state of obtaining the current vehicle state.
- Action:
    - Retrieves information about the current state of the vehicle.
- Transition:
    - Leads to the PredictAction state.

PredictAction State:

- Represents the state where the system predicts the next action based on captured images and the vehicle state.
- Action:
    - Utilizes captured images and the current vehicle state to predict the next action.
- Transition:
    - Leads to the VirtualController state.

VirtualController State:

- Represents the state where the system converts predicted actions into vehicle input.
- Actions:
    - Converts the predicted action to vehicle input.
    - Sends the input to CarDrive.
- Transition:
    - Leads to the CarDrive state.

CarDrive State:

- Represents the state of driving the car based on the processed input.
- Transition:
    - Leads back to the [*] (end) state, indicating the completion of the process.

Transitions:

- The transitions depict the flow of actions and states in the autonomous driving process within the GTA environment.

This state diagram provides a visual representation of the sequential steps involved in preparing a car for autonomous driving and executing actions in a virtual environment.

4) Sequence Diagram Analysis:

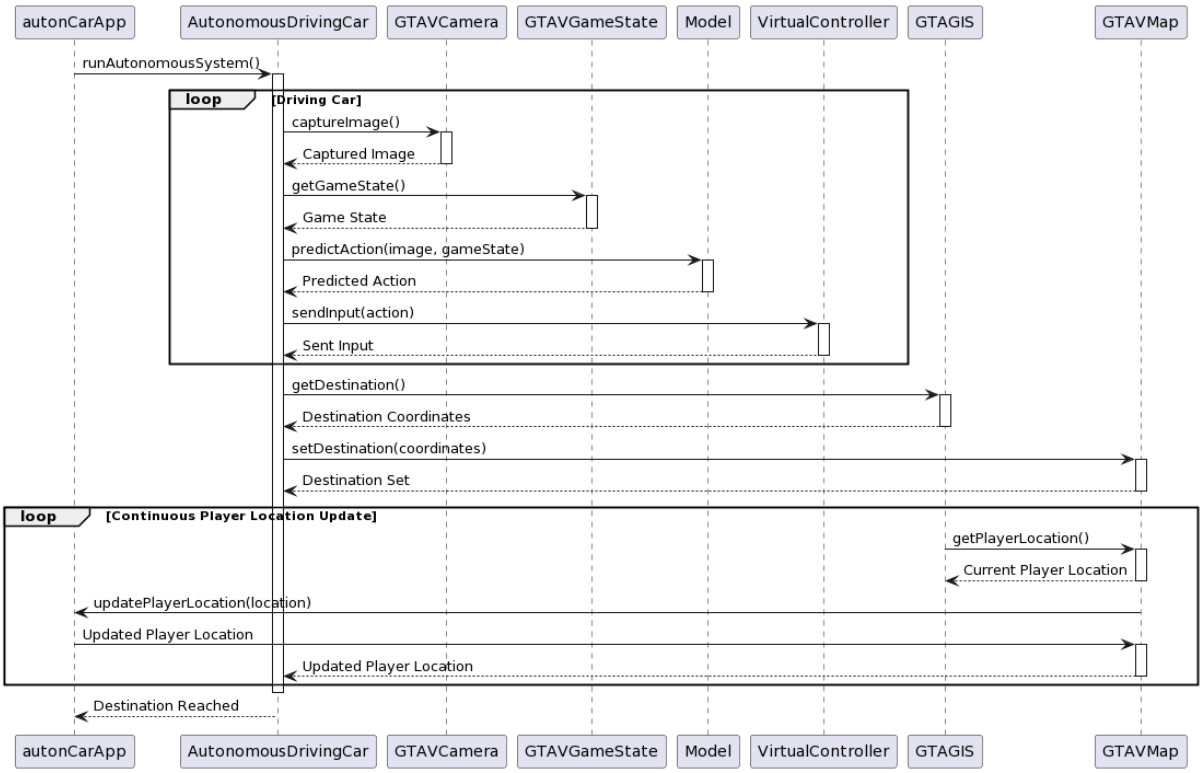


Figure 5: Sequence Diagram of ADV

This sequence diagram illustrates the interactions between components in an ADV application.

Initialization:

- The `autonCarApp` initiates the autonomous driving system by calling the `runAutonomousSystem()` method on the `AutonomousDrivingCar`.
- `AutonomousDrivingCar` becomes active.

Image Capture:

- The `AutonomousDrivingCar` activates the `GTAVCamera` to capture an image.
- `GTAVCamera` captures the image and sends it back to the `AutonomousDrivingCar`.

Game State Retrieval:

- The `AutonomousDrivingCar` retrieves the game state by activating the `GTAVGameState`.
- `GTAVGameState` provides the game state information to the `AutonomousDrivingCar`.

Action Prediction:

- The `AutonomousDrivingCar` activates the `Model` to predict the next action based on the captured image and game state.
- `Model` processes the data and returns the predicted action to the `AutonomousDrivingCar`.

Input Sending:

- The `AutonomousDrivingCar` sends the predicted action to the `VirtualController` for controlling the virtual vehicle.
- `VirtualController` processes the input and sends confirmation back to the `AutonomousDrivingCar`.

Destination Handling:

- The `AutonomousDrivingCar` interacts with the `GTAGIS` to get the destination coordinates.
- It then sets the destination using the `GTAVMap`.

Continuous Player Location Update (Loop):

- The `GTAGIS` queries the `GTAVMap` for the current player location, updating the system continuously.
- The updated player location is sent back to the `autonCarApp`.

Destination Reached:

- The `AutonomousDrivingCar` deactivates, indicating the completion of the autonomous driving system.
- The `autonCarApp` receives the signal that the system has completed.

This sequence diagram demonstrates the flow of actions, communication, and interactions among various components in the ADV application during its operation.

5) Activity Diagram Analysis:

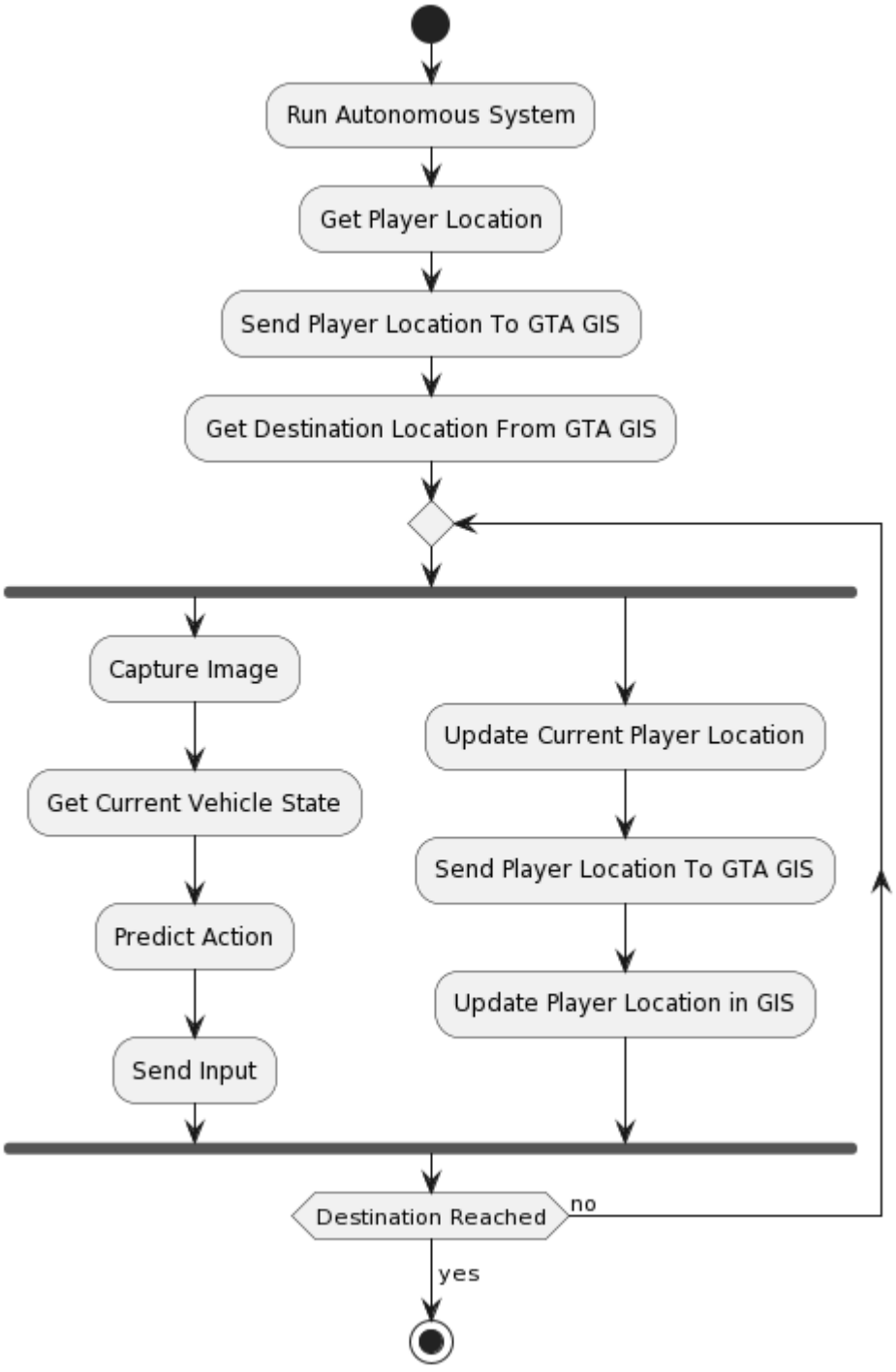


Figure 6: Activity Diagram of ADV

This activity diagram outlines the steps involved in running the Autonomous Driving System within the GTA V environment.

Starting the System:

- The process begins with initiating the Autonomous System.

Getting Player Location:

- The system retrieves the current location of the player within the game.

Communication with GTA GIS:

- The player's location is sent to the GTA GIS (Geographic Information System).

Getting Destination Location:

- The system receives the destination location from the GTA GIS.

Repeating Activities:

- The diagram enters a repeating loop representing the continuous activities of the Autonomous Driving System.

First Fork (Parallel Activities):

- Capturing Image:
    - The system captures images from the virtual environment.
- Getting Current Vehicle State:
    - Information about the current state of the virtual vehicle is obtained.
- Predicting Action:
    - The captured images and vehicle state are used to predict the next action.
- Sending Input:
    - The predicted action is sent as input to control the virtual vehicle.

Second Fork (Parallel Activities):

- Updating Current Player Location:
    - The player's location is continuously updated.
- Communicating with GTA GIS:
    - The updated player location is sent to the GTA GIS.
- Updating Player Location in GIS:
    - The GIS system is updated with the current player location.

Merging Forks:

- The two sets of parallel activities (capturing and updating) converge at this point.

Checking Destination Reached:

- The system checks if the destination is reached.

Repeat or Stop:

- If the destination is not reached, the activities continue in the loop. If the destination is reached, the process stops.

This activity diagram illustrates the continuous operation of the Autonomous Driving System, involving capturing images, predicting actions, updating player location, and checking for the destination reached within the GTA V environment.

4.1.2 Component Diagram

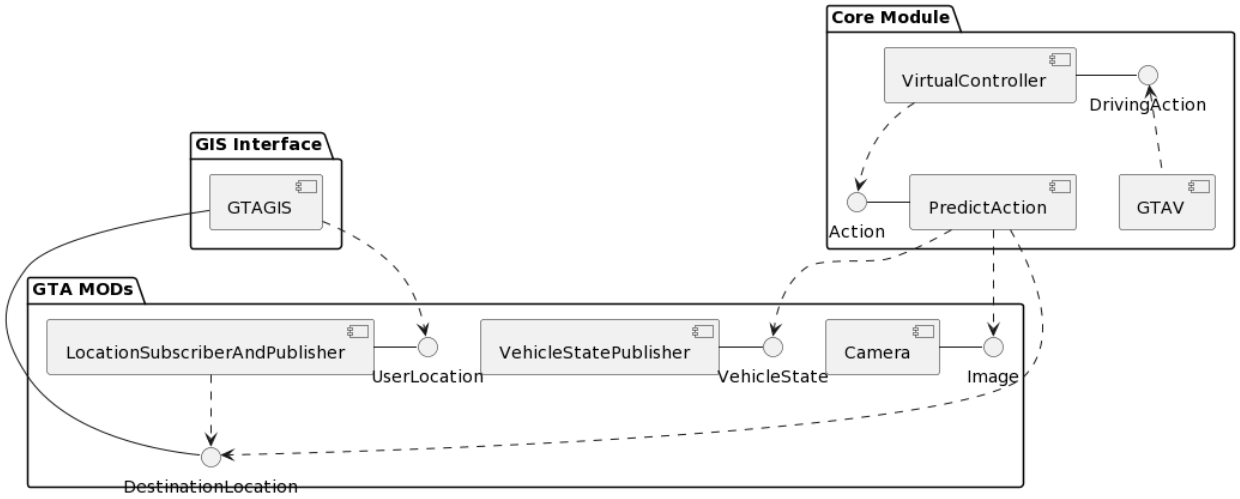


Figure 7: Component Diagram of ADV

This component diagram represents the key components and their interactions in a system designed for the GTA (GTA) environment.

GTA MODs Package:

Camera Component:

- Represents the camera module responsible for capturing images within the GTA environment.
- Produces an Image output.

VehicleStatePublisher Component:

- Represents the module responsible for publishing information about the vehicle state in the GTA environment.
- Produces a VehicleState output.

LocationSubscriberAndPublisher Component:

- Manages the subscription and publication of user and destination locations within the GTA environment.
- Produces UserLocation and DestinationLocation outputs.

GIS Interface Package:

GTAGIS Component:

- Represents the Geographic Information System (GIS) interface, managing destination and user locations.
- Interfaces with UserLocation and DestinationLocation.

Core Module Package:

PredictAction Component:

- Utilizes information from Image, VehicleState, and DestinationLocation to predict the next action.
- Produces an Action output.

VirtualController Component:

- Manages the virtual control of the vehicle based on the predicted action.
- Interfaces with the Action and produces a DrivingAction.

GTA V Component:

- Represents the core module integrating various components for the GTA environment.
- Utilizes and interfaces with the PredictAction and VirtualController components.
- Produces a DrivingAction output.

Interactions:

- The Camera, VehicleStatePublisher, and LocationSubscriberAndPublisher components contribute information to the PredictAction component.

- GTAGIS interfaces with the PredictAction component to provide destination and user location information.
- The PredictAction component interfaces with the VirtualController component to guide the virtual vehicle's actions.
- GTA V represents the central module integrating all components for the GTA environment.

This component diagram provides a visual overview of the modular structure and interactions between components in the system designed for the GTA environment.

4.2 Algorithm Details

The main algorithm that will be used for this project is EfficientNetB1. The use of EfficientNetB1 for ADV has several benefits such as:

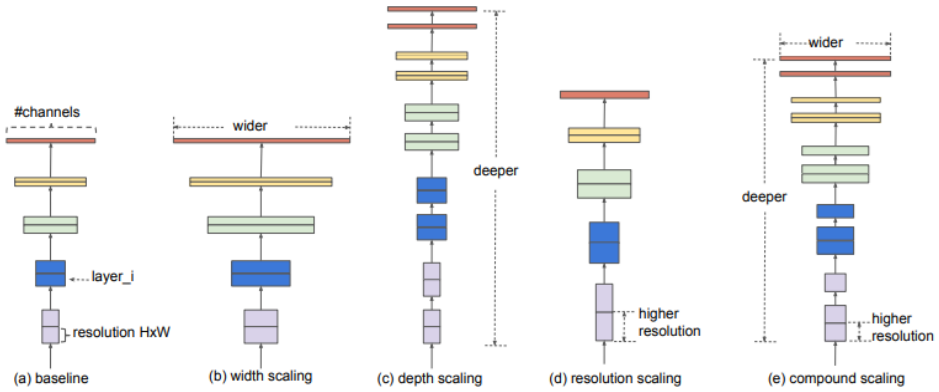
1) **Compound Scaling:** Autonomous driving tasks often require handling complex visual scenes and making decisions based on a wide range of features. EfficientNetB1, with its MBConv block, is capable of learning hierarchical and abstract features, allowing them to capture intricate patterns in visual data.
2) **Feature Learning:** EfficientNets are designed to facilitate the learning of residual connections or "skip connections." These connections allow the model to learn the residual (difference) between the input and output of a layer, making it easier to learn identity mappings and enabling the flow of information across layers. This is especially beneficial for avoiding vanishing gradient problems in deep networks, so helps in creating deeper scaling.
3) **Balanced Scaling:** EfficientNetB1 incorporates a technique called compound scaling, where the model's depth, width, and resolution are scaled together in a principled manner. This ensures that the model maintains a balance between different architectural dimensions, leading to improved performance across various tasks. In the context of autonomous driving, this perfect compound scaling enhances the model's ability to capture intricate details from high-resolution sensor data while maintaining computational efficiency, ultimately contributing to more accurate and reliable decision-making on the road.

4) **Efficient:** As the name suggests, EfficientNet models are designed to achieve better performance while being computationally efficient. As deep neural networks like ResNet have a huge amount of memory requirement, an efficient net helps in this context by being computationally efficient and quick for real-time implementation. For having hardware constraints EfficientNet provides the perfect balance between efficiency and reliability.

EfficientNetB1

EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. Unlike conventional practice that arbitrary scales these factors, the EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. [4]

Model Scaling:



Source: Efficient Net: Rethinking Model Scaling for Convolutional Neural Networks

Figure 8: Model Scaling Example

The model scaling approach proposed in the EfficientNet involves systematically increasing the depth, width, and resolution of convolutional neural networks (CNNs) to improve their performance while maintaining computational efficiency. This approach is based on the observation that scaling these dimensions leads to better model performance. However, simply scaling one dimension without considering the others may not be optimal. Therefore, EfficientNet introduces a compound scaling method that uniformly scales the network depth, width, and resolution using a single scaling parameter. This compound scaling allows for more efficient exploration of the model's architecture space, resulting in

improved performance across various tasks without significantly increasing computational costs.

The model scaling strategy presented in the EfficientNet paper is particularly suitable for self-driving cars due to its balance between performance and computational efficiency. In the context of self-driving cars, real-time processing of sensor data, such as camera images, is crucial for making accurate decisions. The EfficientNet architecture, with its optimized scaling approach, offers a compelling solution by providing high-performance CNN models that can effectively process high-resolution input images while remaining computationally efficient. This enables self-driving car systems to handle complex perception tasks, such as object detection, classification, and semantic segmentation, with minimal computational resources, making them well-suited for deployment in real-world, resource-constrained environments like ADV.

Our model's overall architecture is as follows:

The image input from the vehicle hood camera is fed to EfficientNetB1 which processes the images, after flattening it vehicle state input is added to it and fed to a fully connected neural network for final processing to get the required output of steering angle, throttle power and break.

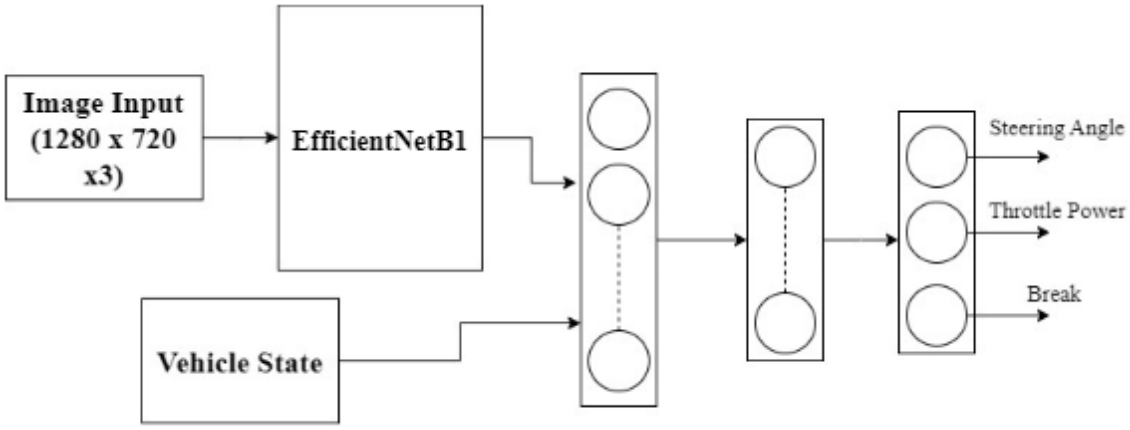


Figure 9: Overall Used Algorithm Architecture of ADV

The activation function used in fully connected neural network's hidden layer is ReLy and for its output layer following activation function is used. The value of the Steering wheel angle can be from -1 to +1 so it will have a linear activation function, the throttle power

will have a value of 0 to +1 so it will have a ReLu activation function, and finally, as the breaking is GTA V is either 0 or 1 so it will have sigmoid activation function.

Algorithm Details:

1. **Convolutional Block:** A convolutional block consists of convolutional layers followed by activation functions. The convolutional operation applies filters to input data to extract features[5].

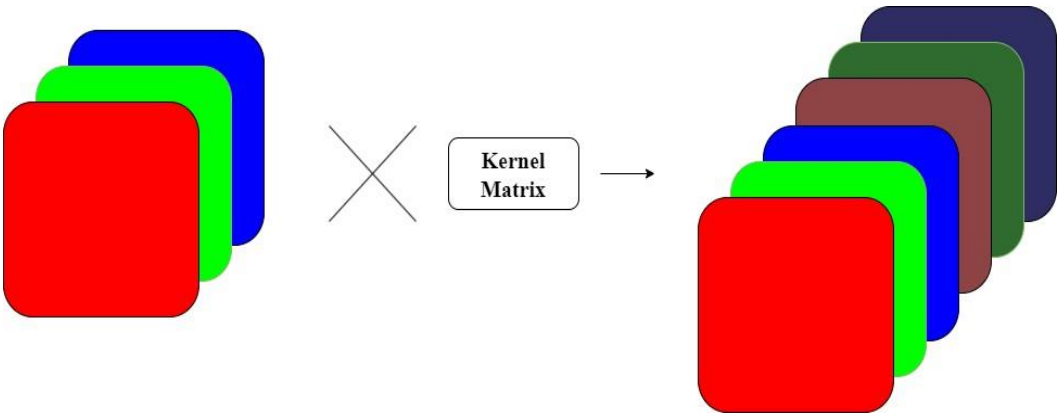


Figure 10: Kernel Matrix Working Example

The Output of the Convolutional block may have a different dimension than the input. We can use different types of kernel for getting different features like vertical line highlighting, horizontal line highlighting, edge highlighting, etc.

2. **Batch Normalization:** Batch Normalization normalizes the input of a layer by adjusting and scaling it to have a mean of 0 and a standard deviation of 1. It helps with training stability and convergence.

$$New\ Value = ((x - \mu) / \sigma) * g + b$$

Where

σ_B is the standard deviation,

μ_B is the Mean,

g, and b are arbitrary parameters that are optimized during the training process.

3. **Max Pooling Block:** Max pooling reduces the spatial dimensions of the input by selecting the maximum value from a group of values in the input[6].

The mathematical equation for Max Pooling is(2D Max Pooling):

$$MaxPooling(x) = max_{ij}\ (x_{i,j})$$

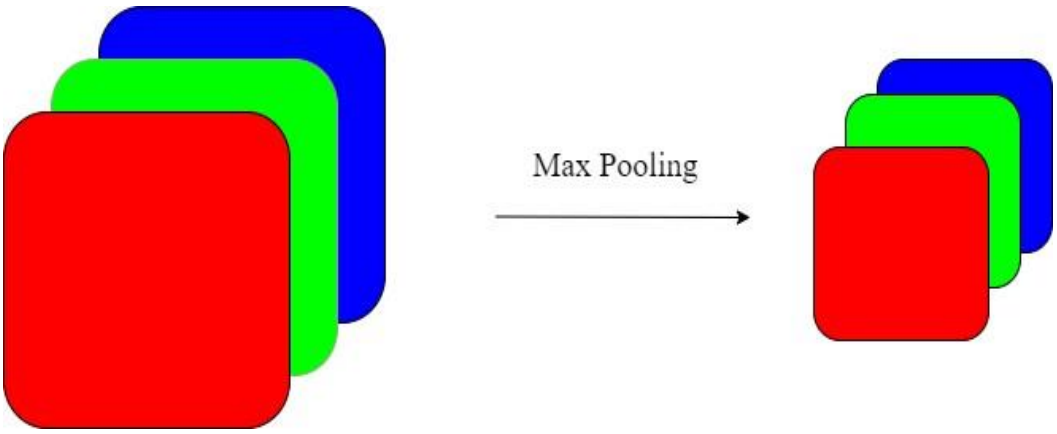


Figure 11: Example Showing Pooling Layer Decreasing Widhth

4. **Residual Block:** A residual block introduces a shortcut connection, allowing the model to learn residual mappings. It helps with training deep networks. Residual Block is added element-wise directly to the output of some convolutional and max pooling blocks to preserve initial features for deep networks.

5. **Fully Connected Neural Network:** A fully connected neural network consists of layers where each neuron is connected to every neuron in the previous and next layers. We are planning to use ReLu for all hidden layers of the fully connected neural network and the output will give us the steering angle, throttle power, and breaking state. The input for the fully connected neural network will be the output of ResNet and vehicle state.

6. **Linear Activation Function:** The linear activation function simply computes a weighted sum of the inputs without introducing non-linearity[7].
   Mathematical Equation:
   $$Linear(x) = w.x + b$$
7. **Sigmoid Activation Function:** The sigmoid activation function squashes input values between 0 and 1, often used for binary classification problems[7]. In our case to get the breaking state.
   Mathematical Equation:
   $$Sigmoid(x) = 1 / (1 + e^{-x})$$

8. **ReLU Activation Function:** The ReLU activation function sets negative values to zero and leaves positive values unchanged, introducing non-linearity[7].
   Mathematical Equation:
   $$ReLU(x) = max(0, x)$$

5. Implementation and Testing

5.1 Implementation

5.1.1. Tools Used

We used the following CASE tools in the implementation of an autonomous vehicle in GTA V:

1) **Visual Studio Code**
   We employed Visual Studio Code for the development of a Geographic Information System (GIS) tailored for the GTA V map. This platform facilitated the integration of HTML, CSS, and JavaScript to enhance mapping functionalities and enable seamless communication with the game.

2) **ChatGPT**
   We utilized ChatGPT as a pivotal resource for addressing queries and resolving uncertainties encountered during the development phase of the Autonomous Driving Vehicle (ADV) system for GTA V. Leveraging its natural language processing capabilities, ChatGPT provided clarity and invaluable insights to streamline project progress.

3) **PlantUML**
   We deployed PlantUML as a Computer-Aided Software Engineering (CASE) tool to visualize and design diverse facets of the ADV system's architecture. Its utilization fostered effective communication and heightened collaborative endeavors among project stakeholders.

4) **PyCharm**

PyCharm played a crucial role in the development of the ADV framework within GTA V. We relied on its proficiency in Python programming to facilitate code composition, organization, and debugging, thereby optimizing the development workflow.

5) **Visual Studio 2022**

A pivotal asset in our development pipeline of the ADV for GTA V, Visual Studio 2022 was primarily employed for scripting in C# to create modifications enriching the system's functionalities within the game environment. Its features facilitated efficient coding, debugging, and overall project management.

6) **Chrome Developer Tool**

Integral to our testing and debugging processes associated with HTML, CSS, and JavaScript implementations for the GIS in GTA V. We relied on this suite of web development tools to ensure seamless integration and provide invaluable insights during the testing phase.

7) **OpenIV**

We relied on OpenIV as a critical tool for integrating external modifications to enhance the ADV system's capabilities within GTA V. Its functionality streamlined the incorporation process, fostering continual enhancement and customization of the project.

8) **ScriptHook**

Essential for implementing bespoke scripts to augment the functionality of the ADV system in GTA V. We utilized this tool to enable the extraction of car state data and facilitate the creation of a tailored camera system to enhance user experience.

5.1.2 Implementation Details of Modules

1) **Game Modifications:**

Car State Mod stands out by capturing and broadcasting a vehicle's dynamic parameters such as throttle power, speed, braking power, and steering angle. Leveraging the NetMQ library, this mod serializes these parameters into JSON

format and transmits them over a local network. This enables external applications to access and utilize vehicle data in real time, enhancing gameplay interactivity and opening up possibilities for detailed game analytics or integration with external devices for a more immersive experience.

Similarly, the Player Position Mod takes immersion a step further by meticulously tracking and sharing the player's location within the vast world of GTA V. By sending the player's coordinates over the network, this mod facilitates the creation of real-time maps or integration with geographic information systems (GIS), allowing players and viewers alike to visualize movements and strategies live. This capability enriches the gaming experience, providing a layer of connectivity and interaction that bridges the game world with real-world applications and technologies.

On the visual front, the Custom Camera Mod revolutionizes the player's visual engagement with the game through dynamic field-of-view adjustments and camera manipulation. By utilizing GTA V's native scripting capabilities, this mod allows players to customize their visual perspective, offering a more tailored and immersive viewing experience. Whether enhancing the depth of exploration or providing strategic visual advantages, this mod adds a valuable layer of customization to the gaming experience, demonstrating the transformative potential of modding on game aesthetics and player immersion.

Collectively, these mods showcase the innovative integration of real-time data exchange and customization within the gaming environment of GTA V. They not only elevate the gameplay experience through enhanced control and visualization but also highlight the potential of game mods to bridge the gap between traditional gaming and advanced interactive technologies. By employing robust tools like NetMQ for messaging and Newtonsoft.Json for data serialization, these mods exemplify a sophisticated approach to game modding that could inspire future developments across various gaming platforms and genres, pushing the boundaries of what is possible in the realm of interactive entertainment.

2) **GTA-GIS:**

The GTA GIS project connects Grand Theft Auto V (GTA V) with a geographic information system (GIS), allowing real-time data exchange and interactive mapping. A real-time server, built using socket.io and aiohttp, acts as a bridge for WebSocket communications. It handles connections from two client types: one sends the player's in-game location, and the other receives updates to display them on a custom web-based map. This setup enables dynamic data exchange, updating player locations on the web map and allowing users to set waypoints that affect in-game navigation.

The map's interface is made with HTML, CSS, and JavaScript, using the Leaflet.js library for map rendering. Users can see their real-time position in GTA V and set destinations or waypoints on the map. Waypoints are sent back to the game through a robust backend setup using ZeroMQ for messaging between the game and server. Python scripts handle this communication, publishing the player's location to the server and subscribing to waypoint data to convey it into the game. This setup creates an interaction loop between the game and GIS, enabling real-time player tracking on the web map and interactive waypoint navigation influenced by user input. This integration showcases the potential for real-time interaction between virtual environments and GIS technologies, enhancing gaming experiences and enabling innovative applications in virtual spatial analysis.

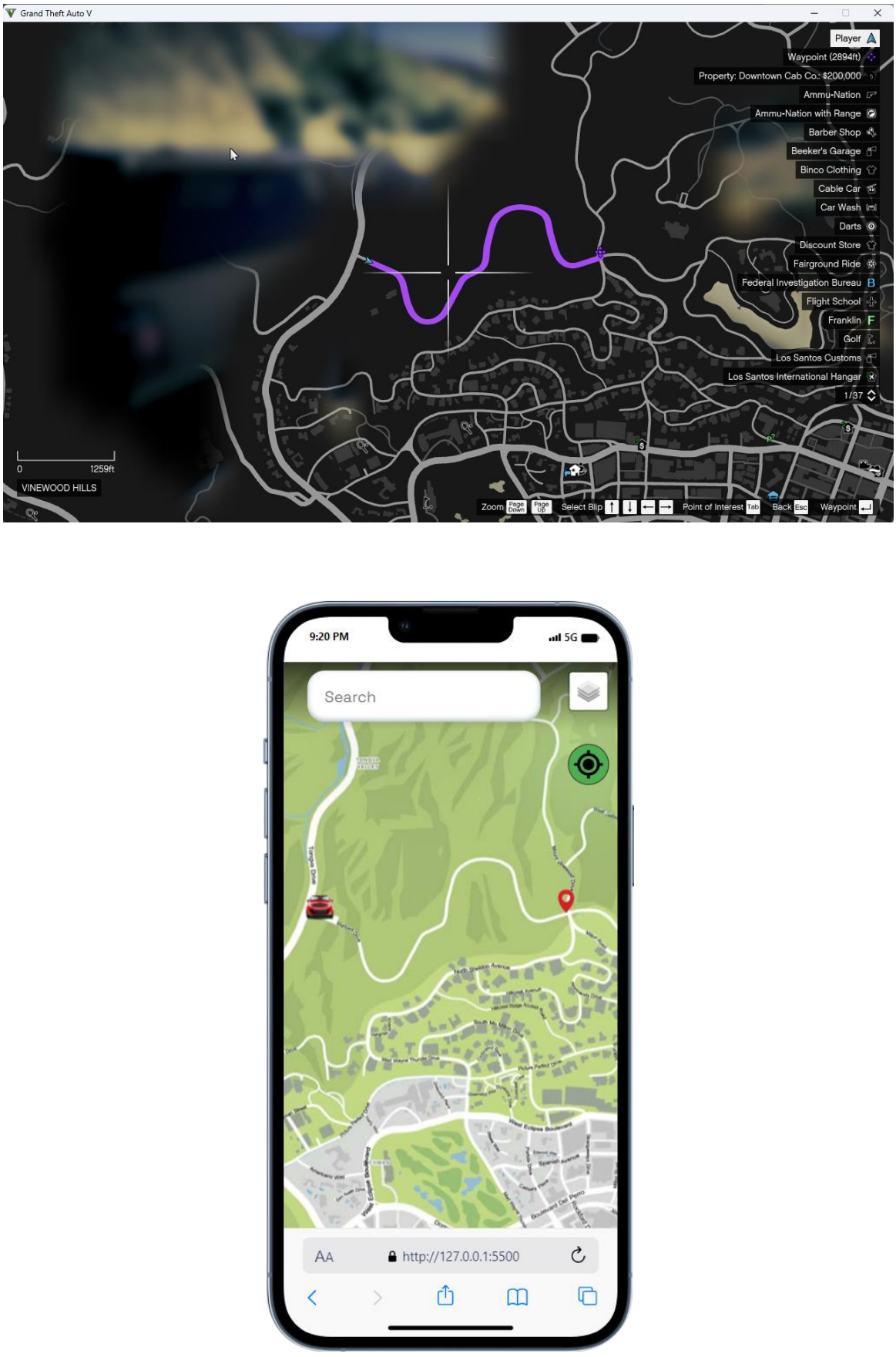


Figure 12: GTA GIS Working

3) **ADV Model:**

The project leverages the capabilities of EfficientNetB1, a cutting-edge convolutional neural network (CNN) architecture known for its efficiency and scalability, to capture and analyze screen content in real time for predictive modeling. This approach involves a sophisticated pipeline where the screen capture serves as the primary data input, which is then processed and fed into the EfficientNetB1 model for analysis. The choice of EfficientNetB1 is strategic, capitalizing on its ability to achieve higher accuracy with fewer parameters compared to other models, thus ensuring both speed and efficiency in real-time applications.

The operational framework begins with the continuous capture of the screen output using a dedicated screen capture tool that is optimized for minimal latency. These captures are then pre-processed to match the input requirements of the EfficientNetB1 model involving resizing and normalization steps. Once pre-processed, the data is fed into the EfficientNetB1 model.

Upon receiving the screen capture data, the EfficientNetB1 model quickly processes it, leveraging its depthwise separable convolutions and compound scaling method to efficiently handle the computations. After the computation of EfficientNetB1, the output is flattened and sent to a fully connected neural along with vehicle state input. The predictions made by the fully connected network are then post-processed and interpreted to provide proper action to send to the virtual controller. The entire process, from screen capture to prediction, is designed to occur in near real-time, enabling dynamic and responsive interactions based on the visual data captured from the screen.

5.2 Testing

To evaluate our system we implemented different testing methods to evaluate our system and determine whether it meets the specified requirements and functions as intended.

5.2.1. Test Cases for Unit Testing

1) **Game State Test:**

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/ Fail)	Notes
1	Accelerating vehicle	Input key for forward movement press.	The acceleration value increases as the key is pressed.	The acceleration value increased as the key was pressed.	Pass	
2	Decelerating vehicle	Input key for forward movement release.	The acceleration value decreases as the key is released.	The acceleration value decreases as the key was pressed.	pass	
3	Breaking vehicle	Input key for breaking press.	The breaking state is 1.	The breaking state is 1.	Pass	
4	Breaking vehicle	Input key for breaking release.	The breaking state is 0.	The breaking state is 0.	Pass	
5	Steering vehicle	Input key for steering press.	Steering angle values changing between -1.0 to +1.0	The steering angle value changed between -1.0 to +1.0	Pass	
6	Steering vehicle	Input key for steering release.	Steering angle values change to 0.	The steering angle value changed to 0.	Pass	

2) **GIS marking and GTA map coordinate testing:**

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/ Fail)	Notes

1	Setting destination mark	Destinations coordinates	Destination mark in GIS.	Destination marked in GIS.	Pass	
2	Setting destination marker in GTA V.	Coordinates from GIS.	Same destination marker in GIS and GTA V.	The same destination is marked in GIS and GTA V.	Pass	
3	Destination reached notification	Distance from vehicles position to destination marker	System triggers a notification alert to inform user for four seconds	The notification alert displayed for four seconds but occasionally blinked due to coordinate changes.	Fail	Alert box blinks sometimes

5.2.2 Test Cases for System Testing

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/ Fail)	Notes
1	Set destination marker in GIS and press I in GTAV to set destination marker.	Destination coordinates	Destination mark sets in the in-game GTAV map.	Destination marker set in the in-game GTAV map.	Pass	
2	Run the GTAV game, ADV_model.	Screencapture of GTAV to model and game control inputs as input to GTAV from the model.	Vehicle drives autonomously.	The vehicle drove autonomously.	Pass	

6. Conclusion and Future Recommendations

6.1 Conclusion

In conclusion, our project has effectively shown how we can achieve Level 2 self-driving capabilities inside the game world of Grand Theft Auto V (GTA V), using the power of EfficientNetB2 for analyzing images and making decisions instantly. We've put together a system that moves through the game's changing world and also lets users direct the car using a simple and easy-to-use graphical user interface. This GUI was made with modern web tools like Leaflet.js for showing maps, socket.io, and aiohttp for fast server communication, allowing users to easily choose where the car should go and interact with it straightforwardly.

Reaching Level 2 autonomy in a game setting like GTA V is a big deal for us, showing how we can mix advanced learning models like EfficientNetB2 with complex web and server tech to make self-driving experiences that are both deep and interactive. Also, making a friendly interface for users to pick destinations in real time highlights our effort to make self-driving tech more user-friendly. This project moves the virtual self-driving field forward and sets the stage for more work in bringing together AI, interactive mapping, and quick data sharing for even smarter self-driving systems.

6.2 Future Recommendations

To advance the project further, we recommend focusing on integrating advanced lane detection and object recognition technologies. By applying deep learning models for precise lane segmentation, the autonomous vehicle can navigate with greater accuracy, maintaining lane discipline and adjusting its road position effectively. Additionally, incorporating object detection for traffic lights, pedestrians, and traffic signs will enable the vehicle to interact safely and intelligently with its surroundings. This step is crucial for recognizing and responding to traffic cues, ensuring pedestrian safety, and adhering to road regulations, thereby elevating the system's overall safety and performance.

Leveraging the enhanced capabilities of improved lane detection and object recognition will significantly refine the vehicle's decision-making processes. The integration of comprehensive environmental data allows for more sophisticated navigation strategies, facilitating the vehicle's ability to handle complex driving scenarios such as intersections

and emergency situations. These advancements will not only augment the autonomous driving experience within the virtual world of GTA V but also contribute insights into real-world autonomous vehicle technologies, marking a step forward in achieving higher levels of driving autonomy and simulation realism.

References

[1] Waymo. [Online]. Available: https://waymo.com/ [Accessed: Sep. 24, 2023].

[2] S. Behzad, "Game Frames OpenCV - Python Plays GTA V," [Online]. Available: https://pythonprogramming.net/game-frames-open-cv-python-plays-gta-v/ [Accessed: Sep. 28, 2023].

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," [Online]. Available: https://www.cs.princeton.edu/research/techreps/TR-005-18 [Accessed: Oct. 15, 2023].

[4]“EfficientNet,” Papers With Code. [Online]. Available: https://paperswithcode.com/method/efficientnet#:~:text=EfficientNet%20is%20a%20convolutional%20neural,resolution%20using%20a%20compound%20coefficient. [Accessed: Feb. 24, 2024].

[5]H. S. Chatterjee, "A Basic Introduction to Convolutional Neural Network," Medium, [Online].Available:https://medium.com/@himadrisankarchatterjee/a-basic-introduction-to-convolutional-neural-network-8e39019b27c4 [Accessed: Nov. 4, 2023].

[6]"Papers with Code: Max Pooling," Papers with Code, [Online]. Available: https://paperswithcode.com/method/max-pooling [Accessed: Nov. 4, 2023].

[7]"Activation Functions in Neural Networks," Towards Data Science, [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 [Accessed: Nov. 18, 2023].