Dharmsinh Desai University, Nadiad

Faculty of Technology

Department of Computer Engineering
B. Tech. CE Semester – VI

Subject:SYSTEM DESIGN PRACTICE (MINI PROJECT)

Project Title:V-ADAS

Guided by:

Prof. Jignesh k. Shah

Assistant Professor

Dharmsinh Desai University
Faculty of Technology, College Road, Nadiad – 387001, Gujarat

CERTIFICATE

This is to certify that the term work carried out in the subject of SYSTEM DESIGN PRACTICE (MINI PROJECT) and submitted is the bonafide work of **Gajera Kavan S** Roll No.: **CE-106** Identity No.: **22CEUOS088** and **Madariya Kanji** Roll No.: **CE-127** Identity No.: **22CEUOS117** **of** B.Tech. **Semester VI** in the branch of Computer Engineering during the academic year **2024-2025.**

Prof. Jignesh Shah	Dr. C. K. Bhensdadia
Assistant Professor	Head of Department
CE Department	CE Department

Table of contents

Introduction to Advanced Driver Assistance Systems (ADAS)

What is ADAS?

Advanced Driver Assistance Systems (ADAS) are intelligent solutions that enhance the safety and convenience of driving. Utilizing cutting-edge technologies like sensors, cameras, radars, and artificial intelligence (AI), ADAS provides real-time insights and proactive assistance to drivers. From preventing collisions to enabling semi-autonomous driving, these systems are revolutionizing the automotive industry.

Real-World Impact of ADAS

Imagine driving on a rainy night when visibility is low. Your car's Adaptive Cruise Control (ACC) maintains a safe distance from the vehicle ahead, while Lane Keeping Assist (LKA) gently nudges the steering to keep you within your lane. If a pedestrian suddenly appears, the Automatic Emergency Braking (AEB) system instantly applies the brakes, preventing an accident. This is the power of ADAS in action.

Popular ADAS Features in Vehicles Today

- **Adaptive Cruise Control (ACC)**: Adjusts your speed to maintain a safe following distance.
- **Lane Departure Warning (LDW) and Lane Keeping Assist (LKA)**: Alerts and corrects lane drifts.
- **Automatic Emergency Braking (AEB)**: Prevents collisions by applying brakes when necessary.
- **Blind Spot Detection (BSD)**: Alerts drivers to vehicles in their blind spots.
- **Parking Assistance**: Provides automated steering or visual guidance while parking.

The Role of Simulators in ADAS Development

Developing and testing ADAS features in real-world environments can be expensive and risky. This is where simulators like **Grand Theft Auto V (GTA V)** come into play. With its realistic graphics and dynamic scenarios, GTA V serves as an excellent virtual testing ground for ADAS algorithms. Using Python-based AI models, developers can simulate complex driving scenarios, fine-tune system responses, and ensure robust performance before real-world deployment.

Conclusion

ADAS is not just a technological advancement; it's a step toward safer roads and smarter vehicles. By leveraging powerful simulations and machine learning algorithms, the future of autonomous driving is closer than ever. In the upcoming sections, we will dive deeper into the design and implementation of our ADAS system, highlighting how we harnessed Python and GTA V for this innovative project.

Literature Review

Pioneers of Autonomous Driving

The journey toward autonomous driving has been marked by significant contributions from various innovators and organizations. Early experiments in robotics and artificial intelligence laid the groundwork, with notable advancements emerging in the late 20th and early 21st centuries.

Tesla's Impact on ADAS

Tesla has been instrumental in bringing Advanced Driver Assistance Systems (ADAS) to the forefront of consumer vehicles. Through its **Autopilot** and **Full Self-Driving (FSD)** systems, Tesla integrates cameras, ultrasonic sensors, and AI models to enable features like highway navigation, lane changes, and obstacle avoidance. The company's commitment to over-the-air updates ensures continuous enhancement of its autonomous capabilities.

Waymo's Innovations in Autonomous Driving

Waymo, a subsidiary of Alphabet, has been a trailblazer in self-driving technology. With extensive real-world testing and advanced sensor fusion techniques, Waymo has developed a fleet of autonomous taxis operating in multiple cities, demonstrating the practicality of fully autonomous transportation.

Contributions from Independent Creators like Snatex

Independent developers have also made significant strides in ADAS research. Creators like **Snatex** have utilized simulators such as **GTA V** to test and refine autonomous driving algorithms. By leveraging Python and machine learning frameworks, Snatex's work exemplifies the potential of virtual environments in advancing autonomous driving technology.

Evolution of Key Technologies in ADAS

The Invention of RADAR

The principles of **RADAR (Radio Detection and Ranging)** date back to the late 19th century. In 1886, Heinrich Hertz demonstrated that radio waves could be reflected from solid objects, laying the foundation for radar technology. The first patent for a radar-like device was filed in 1904 by German engineer **Christian Hülsmeyer**, who developed the "Telemobiloskop" to detect ships and prevent collisions.

The Development of LiDAR

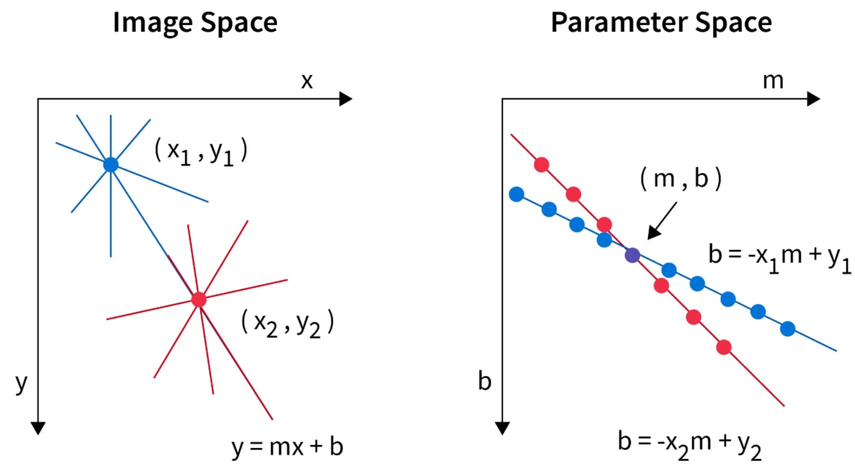
LiDAR (Light Detection and Ranging) technology emerged in the 1960s, shortly after the invention of the laser. Initially used for meteorological applications, LiDAR has since become integral to autonomous vehicles, enabling precise 3D mapping of environments.

Advancements in Image Processing Algorithms

Image processing plays a crucial role in ADAS, with several key algorithms contributing to the detection and interpretation of visual information.

The Hough Transform

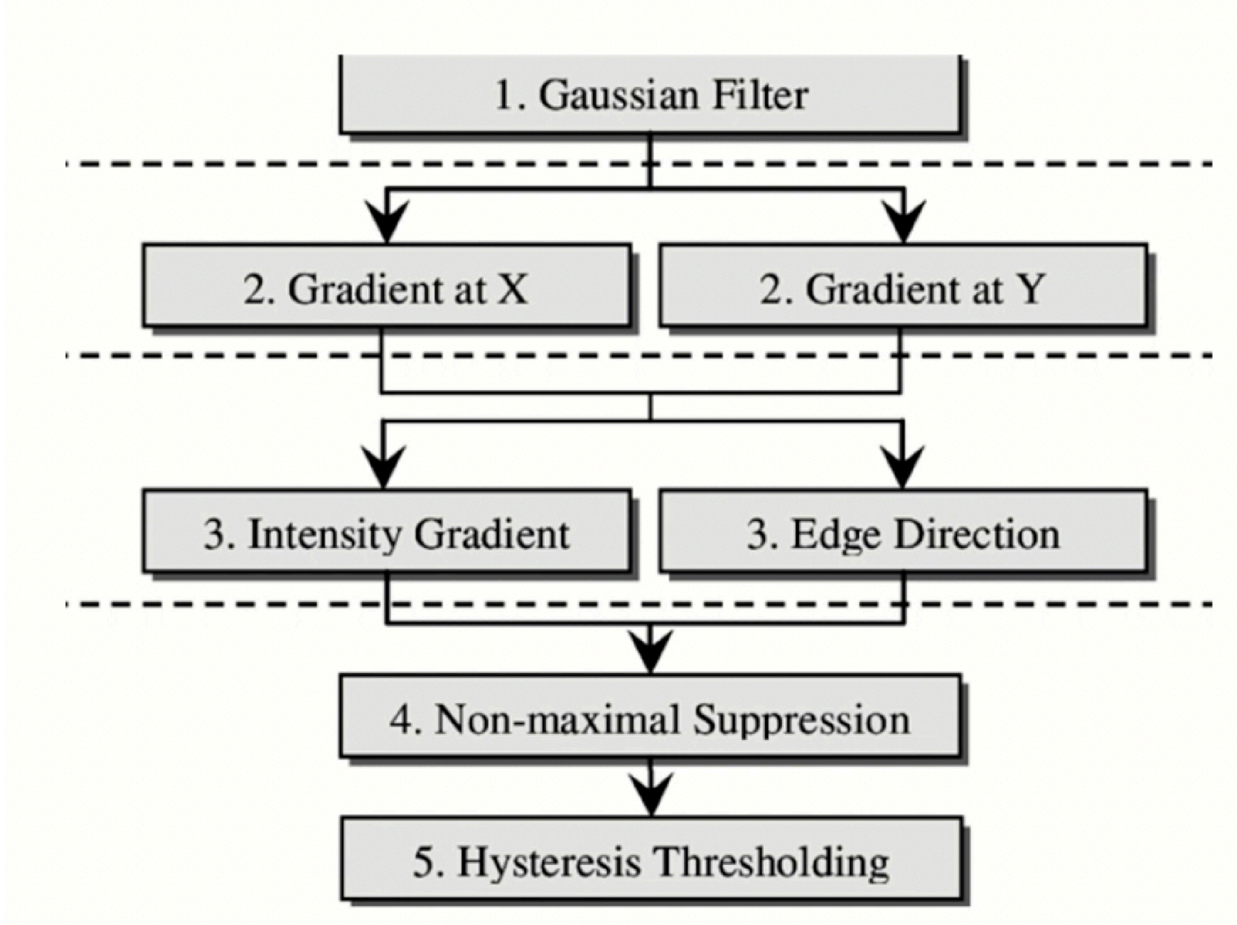
Introduced by **Paul Hough** in 1962, the Hough Transform is a technique for detecting shapes, particularly lines, within images. Originally developed for analyzing bubble chamber photographs in particle physics, it has since been widely adopted in computer vision for feature extraction.



SCALER Topics

Canny Edge Detection

Developed by **John F. Canny** in 1986, the Canny Edge Detection algorithm aims to identify edges in images with optimal accuracy. Canny's approach involved a comprehensive set of goals to enhance edge detection performance, making it a foundational tool in image processing.



Conclusion

The evolution of autonomous driving and ADAS is the result of collaborative efforts spanning over a century, involving pioneering scientists, innovative companies, and independent researchers. From the early developments of RADAR and LiDAR to sophisticated image processing algorithms, each advancement has contributed to the realization of safer and more efficient transportation systems.

Algorithms and Models

In the development of your Advanced Driver Assistance System (ADAS) using Python and GTA V as a simulator, several key algorithms and models play pivotal roles. Below is an in-depth overview of these components, including their mathematical foundations and practical applications.

Hough Line Transform

Overview

The **Hough Line Transform** is a fundamental technique in image processing used to detect straight lines within an image. By transforming points from the Cartesian coordinate system to a parameter space, it identifies lines even in noisy environments.

Mathematical Foundation

In the Cartesian coordinate system, a straight line is represented as:

$$y = mx + c$$

Where:

- `m` is the slope,
- `c` is the y-intercept.

The Hough Transform, however, utilizes the polar coordinate representation:

$\rho = x * \cos(\theta) + y * \sin(\theta)$

Here:

- ρ is the perpendicular distance from the origin to the line,
- θ is the angle between the x-axis and the line connecting the origin to the closest point on the line.

By mapping all edge points (x, y) in the image to the (ρ, θ) space, the algorithm identifies the accumulation points corresponding to potential lines.

Application

1. **Edge Detection**: Apply an edge detection algorithm (e.g., Canny Edge Detector) to identify potential edge points.
2. **Parameter Space Mapping**: For each edge point, compute possible (ρ, θ) values and map them in an accumulator matrix.
3. **Peak Detection**: Identify peaks in the accumulator matrix, which correspond to the most likely lines in the image.

For a comprehensive guide on the Hough Transform, refer to this resource.

Canny Edge Detection

Overview

Developed by **John F. Canny** in 1986, the Canny Edge Detection algorithm is renowned for its ability to detect a wide range of edges in images with optimal accuracy.

Process

The algorithm involves several steps:

1. **Noise Reduction**: Apply a Gaussian filter to smooth the image and reduce noise.

2. **Gradient Calculation**: Compute the intensity gradients of the image.

3. **Non-Maximum Suppression**: Thin the edges by suppressing pixels that are not at the maximum gradient.

4. **Double Thresholding**: Classify edges as strong, weak, or non-relevant based on gradient magnitude thresholds.

5. **Edge Tracking by Hysteresis**: Finalize edge detection by connecting weak edges to strong edges if they are in close proximity.

Mathematical Foundation

Gaussian Smoothing:

$$H(i, j) = (1 / (2\pi\sigma^2)) * \exp(-((i - (k+1))^2 + (j - (k+1))^2) / (2\sigma^2))$$

1. Where σ is the standard deviation of the Gaussian distribution, and $(2k+1) \times (2k+1)$ is the kernel size.

Gradient Calculation:

$$G = \sqrt{(Gx^2 + Gy^2)}$$

$$\theta = \text{atan2}(Gy, Gx)$$

2. Where Gx and Gy are the gradients in the x and y directions, respectively, and θ is the gradient direction.

For a detailed explanation of the Canny Edge Detector, visit this page.

YOLOv8n for Object Detection

Overview

YOLO (You Only Look Once) is a state-of-the-art, real-time object detection system. The **YOLOv8n** variant is a lightweight model designed for high-speed applications, making it suitable for real-time scenarios like driving simulations.

Architecture

YOLOv8n employs a single neural network that divides the image into a grid and predicts bounding boxes and class probabilities for each grid cell simultaneously. This unified approach allows for rapid and accurate object detection.

Mathematical Foundation

The model optimizes the following multi-part loss function:

Loss = λ_coord * Σ $(x - \hat{x})^2 + (y - \hat{y})^2$ + λ_coord * Σ $(\sqrt{w} - \sqrt{\hat{w}})^2 + (\sqrt{h} - \sqrt{\hat{h}})^2$

+ Σ $(C - \hat{C})^2$ + λ_noobj * Σ $(C - \hat{C})^2$ + Σ_i Σ_c (p_i(c) - p̂_i(c))^2

Where:

- `(x, y, w, h)` represent the bounding box coordinates and dimensions,
- `C` is the confidence score,
- `p_i(c)` is the class probability,
- `λ_coord` and `λ_noobj` are hyperparameters to balance the loss components.

For an in-depth guide on YOLO object detection, refer to this article.

Support Vector Machines (SVM)

Overview

Support Vector Machines are supervised learning models used for classification and regression tasks. They are particularly effective in high-dimensional spaces and are widely applied in pattern recognition.

Mathematical Foundation

Given a training dataset with labels, SVM aims to find the optimal hyperplane that separates the data points of different classes with the maximum margin. The decision function is defined as:

$f(x) = \text{sign}(w \cdot x + b)$

Where:

- `w` is the weight vector,
- `x` is the input feature vector,
- `b` is the bias term.

The optimization problem involves minimizing `||w||²` subject to the constraint that all data points are correctly classified with a margin of at least 1.

Kernel Trick

For non-linearly separable data, SVM employs kernel functions to map the input features into higher-dimensional spaces where a linear separator may exist. Common kernels include:

- **Linear Kernel**: `K(x, y) = x · y`
- **Polynomial Kernel**: `K(x, y) = (x · y + c)^d`

- **Radial Basis Function (RBF) Kernel**: $K(x, y) = \exp(-\gamma ||x - y||^2)$

studied from :

Support Vector Machines : Data Science Concep…

Our Extraction from Literature and Studies

In our project, we are creating a **Virtual Advanced Driver Assistance System (ADAS)**, serving as a prototype using **GTA V** as the simulator. The virtual environment in GTA V offers realistic driving scenarios that are ideal for testing ADAS algorithms.

We utilized **OpenCV** for real-time screen capturing and applied **PyAutoGUI** for vehicle control. The key objectives of our ADAS system include:

- **Minimap Detection**: Extracting and analyzing the in-game minimap to assess traffic, road direction, and surroundings.
- **Lane Detection**: Implementing computer vision algorithms like **Canny Edge Detection** and **Hough Line Transform** for accurate lane detection.
- **Object Detection**: Utilizing state-of-the-art models like **YOLOv8n** to identify vehicles, pedestrians, and obstacles, ensuring safe navigation.

Through this virtual ADAS system, we simulate complex driving scenarios to test lane-keeping, collision avoidance, and traffic management algorithms. The prototype serves as a stepping stone for further development and validation of autonomous driving technology in real-world applications.

Game Data Collection Project Technical Overview

Dataset Details

Data Collection Methodology

- Custom dataset created during gameplay of GTA 5
- Manual screenshot collection and labeling
- Real-time tracking of key presses and car dynamics

Dataset Characteristics

Label	Description	Data Captured
Key Press	Keyboard input tracking	a', 's', 'd' keys
Duration	Time key is pressed	Milliseconds
Car Position	X, Y coordinates	Pixel coordinates
Distance Calculation	Car movement tracking	Focal Length Formula

Focal Length Distance Calculation Formula

Distance = (Actual Object Width * Focal Length) / (Pixel Width)

Labeling Schema

Column	Description	Example Values
Key	Pressed keyboard key	a', 's', 'd'
Duration	Key press time	100 ms
X-Coordinate	Car horizontal position	350 pixels
Y-Coordinate	Car vertical position	240 pixels
Distance	Calculated car distance	5.6 meters

Tools and Technologies

Software Tools

Category	Tools	Purpose
Computer Vision	OpenCV	Image processing, edge detection
Object Detection	YOLOv8	Real-time object recognition
Game Platform	GTA 5	Simulation environment
Development Environments	Google Colab, Jupyter Notebooks	Code development and execution

Python Libraries

Library	Functionality	Key Features
scikit-learn (sklearn)	Machine Learning	SVM Classification
OpenCV	Computer Vision	Image processing, Hough Line detection
PyAutoGUI	GUI Automation	Screen capture, mouse/keyboard control
NumPy	Numerical Computing	Array operations, mathematical functions
Pandas	Data Manipulation	Dataset management, analysis
Matplotlib	Data Visualization	Plotting, graphing results

Technologies and Domains

Domain	Technologies	Application
Programming	Python	Primary development language
Machine Learning	Supervised Learning	SVM Classification
Artificial Intelligence	Computer Vision	Object detection, tracking
Data Processing	Statistical Analysis	Feature extraction, model training

Project Technological Stack

- **Primary Language**: Python
- **Machine Learning**: Supervised Learning (SVM)
- **Computer Vision**: OpenCV, YOLOv8
- **Development Environment**: Google Colab, Jupyter Notebooks
- **Game Simulation**: GTA 5
- **Key Technologies**: AI, Machine Learning, Computer Vision

Implementation Details

Simulator Selection: GTA-5

Justification for Choosing GTA-5

- Vast, realistic open-world environment
- Detailed graphics mimicking real-world driving scenarios
- Comprehensive vehicle dynamics simulation
- Rich, complex urban landscape for comprehensive data collection

Technical Implementation Stages

1. Screen Capture Mechanism

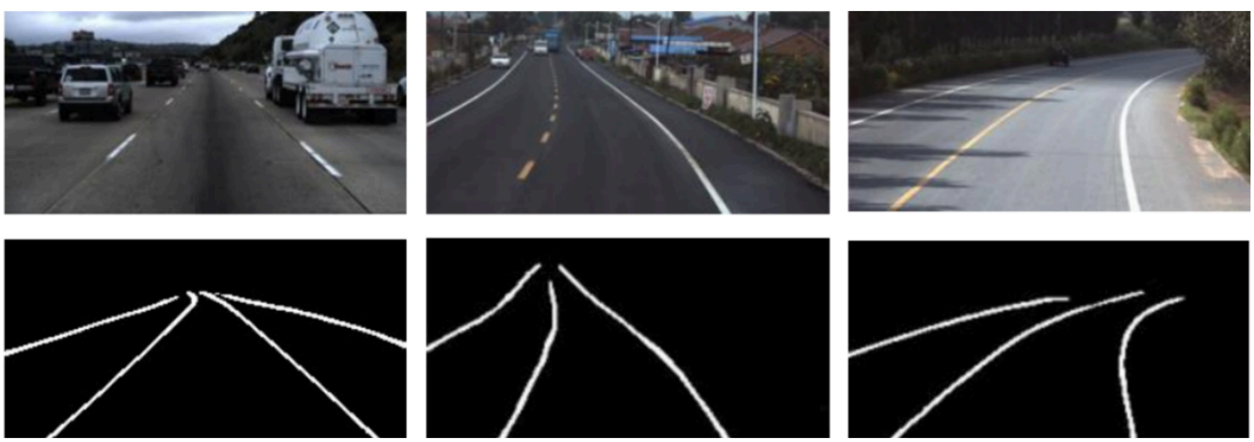
- Utilize OpenCV for high-performance screen recording
- Capture high FPS (Frames Per Second) gameplay footage
- Ensure minimal performance overhead
- Synchronize capture with game rendering

2. Navigation Analysis

Mini Map Detection

- Extract mini map region of interest (ROI)
- Implement image processing techniques
- Analyze driving direction and trajectory
- Convert map coordinates to actionable driving data
- 

3. Lane Detection Process

- Apply computer vision algorithms
- Use Canny edge detection
- Implement Hough Line Transform
- Identify lane boundaries
- Extract lane geometry and curvature
- 

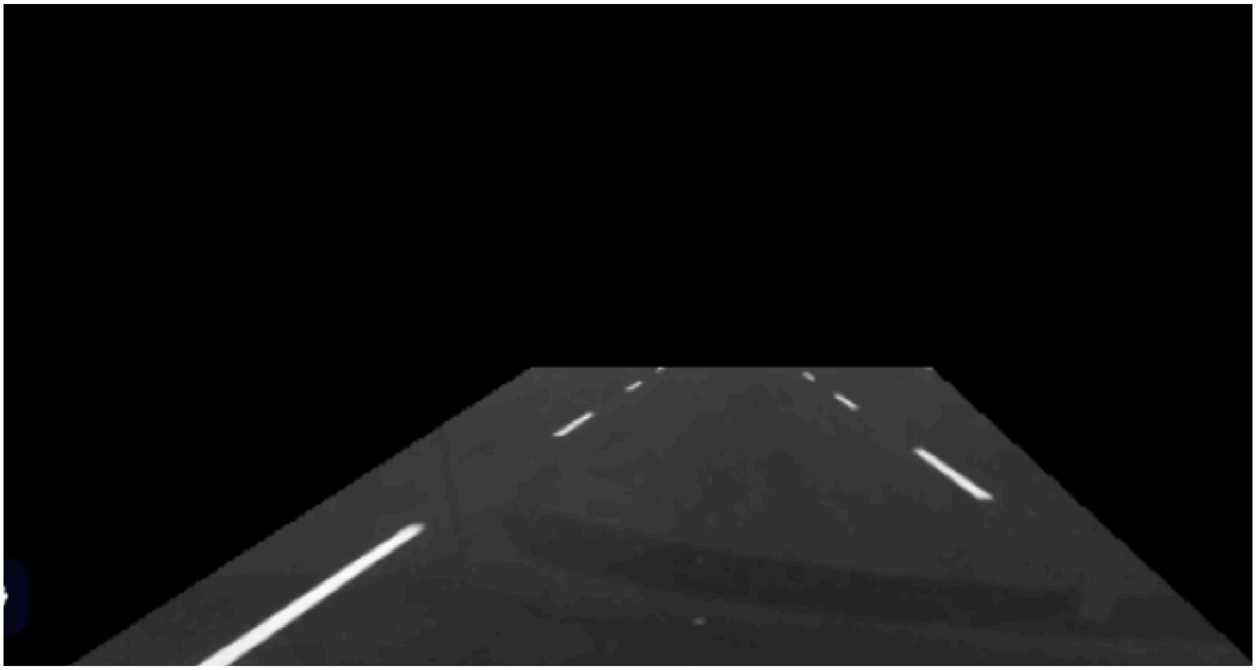
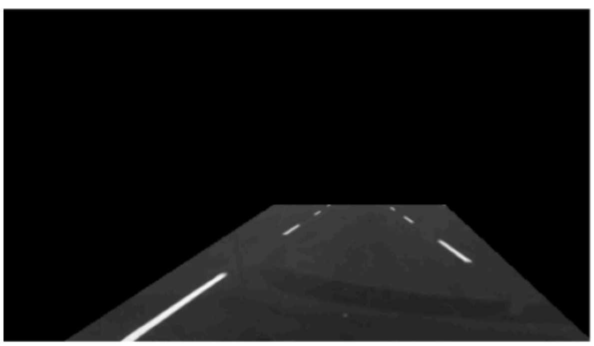
- 
- 
  Masked Image

  
  Image after Thresholding

- 

4. Object Detection

- Implement YOLOv8 for real-time object recognition
- Detect and classify:
  - Other vehicles
  - Pedestrians
  - Traffic signals
  - Road infrastructure


FPS: 0.9
Action: BRAKE (Space)
Prediction: 1
Speed: 0.27
Closest Car: 1.5m
Truck: 1.5m


FPS: 1.9
Action: RIGHT (D)
Prediction: 2
Speed: 0.27
Closest Car: 1.7m
Car: 1.7m
09VYF811
0.60mi


FPS: 11.2
Action: LEFT (A)
Prediction: 0
Speed: 0.27
Closest Car: 2.3m
Person: 2.3m
Person: 4.1m


Warning: Red Light Ahead (21.0m)
TRAFFIC RED (0.46) 21.0m
car 8.34m
car 13.12m
person 31.82m
BRAKE
Light 1: (366,150) red 21.0m conf:0.46
Traffic lights detected: 1/2

5. Dataset Collection Strategy

- Systematic data gathering from gameplay
- Capture key parameters:
  - Keyboard inputs
  - Car coordinates
  - Driving duration
  - Environmental conditions

6. Machine Learning Model Training

Support Vector Machine (SVM) Configuration

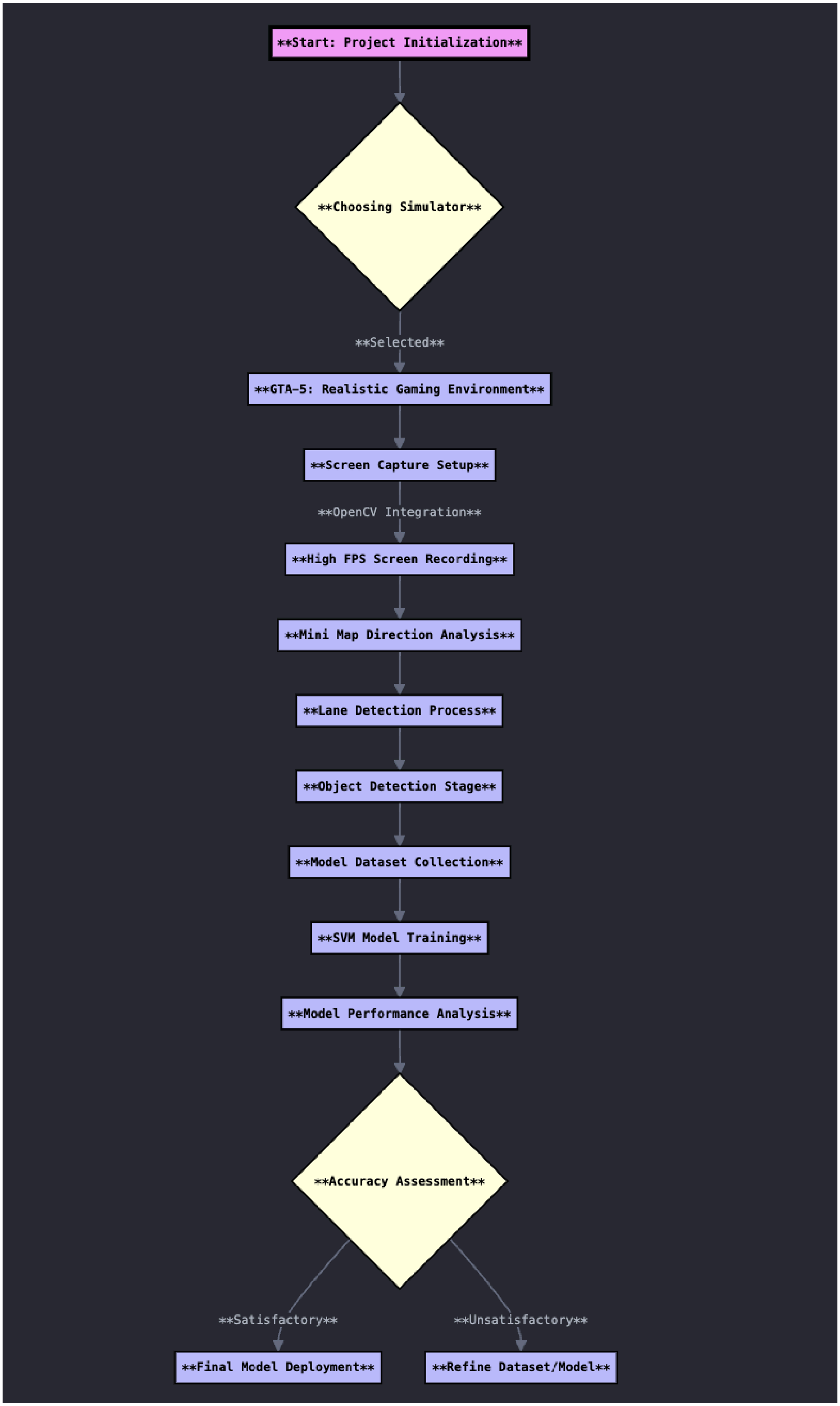
- Feature vector preparation
- Hyperparameter tuning
- Cross-validation techniques
- Margin maximization
- Kernel selection (linear/radial basis function)

7. Model Evaluation

- Accuracy assessment
- Confusion matrix analysis
- Precision and recall metrics
- Datapoint separability evaluation
- Iterative model refinement

Continuous Improvement Approach

- Regular model retraining
- Expanding dataset diversity
- Incorporating feedback mechanisms
- Performance monitoring


Start: Project Initialization
Choosing Simulator
Selected
GTA-5: Realistic Gaming Environment
Screen Capture Setup
OpenCV Integration
High FPS Screen Recording
Mini Map Direction Analysis
Lane Detection Process
Object Detection Stage
Model Dataset Collection
SVM Model Training
Model Performance Analysis
Accuracy Assessment
Satisfactory
Unsatisfactory
Final Model Deployment
Refine Dataset/Model

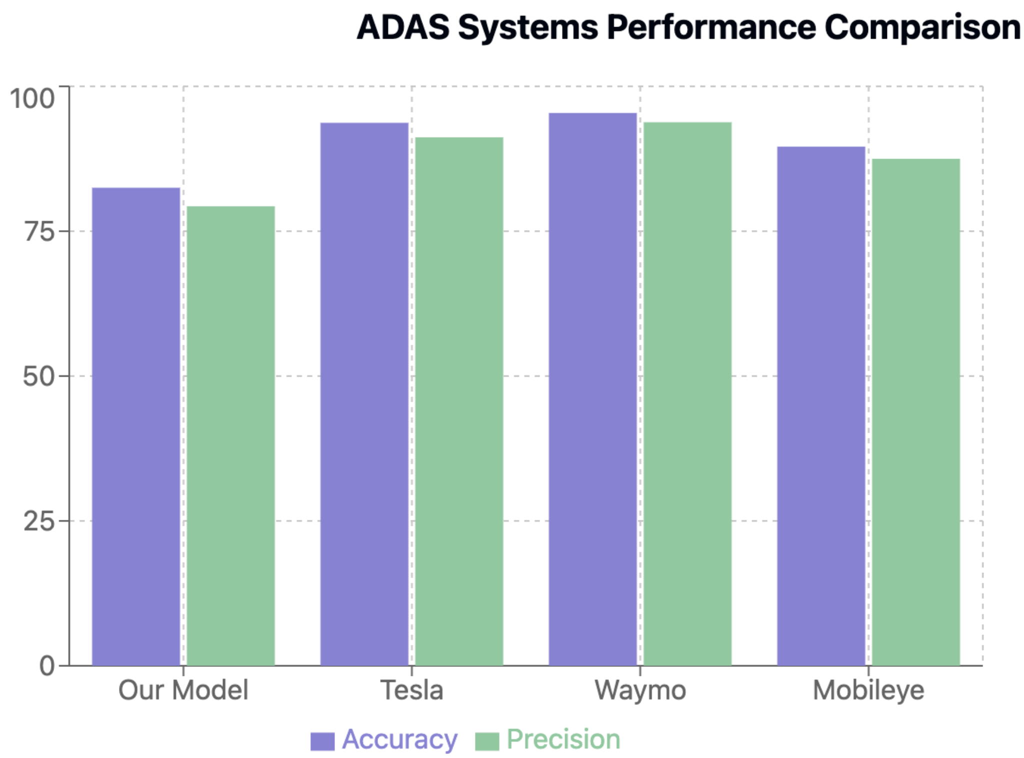
Testing and Results

Comparative Analysis of ADAS Systems

Performance Metrics Comparison

System	Accuracy	Precision	Object Detection	Environment Adaptability
Our SVM Model	82.50%	79.30%	Limited	Urban/Game Environment
Tesla Autopilot	93.70%	91.20%	Advanced	Multiple Terrains
Waymo	95.40%	93.80%	Highly Advanced	Diverse Conditions
Mobileye	89.60%	87.50%	Advanced	Multiple Scenarios

Comparative Visualization



Carla Simulator Testing

Methodology

- Integrated our SVM model with Carla simulator
- Python-based implementation
- Comparative performance analysis
- Scenario-based testing across different urban environments

Testing Scenarios

1. Urban intersection navigation
2. Highway lane keeping
3. Obstacle avoidance
4. Traffic signal response

Conclusion and Future Roadmap

Proposed Enhancements

1. **Enhanced Traffic Light Detection**
   - Advanced computer vision algorithms
   - Multi-state recognition
   - Night and adverse weather conditions support
2. **360-Degree Camera Integration**
   - Multiple camera perspectives
   - Comprehensive environmental awareness
   - Blind spot elimination
3. **Speed Board Analysis**
   - Real-time speed limit recognition
   - Automatic speed adjustment
   - Regulatory compliance

India-Specific Adaptations

- Developed for diverse Indian road conditions
- Handling unique traffic scenarios
- Supporting local traffic patterns
- Multilane and mixed-traffic environment optimization

🇮🇳 India-Focused Development 🇮🇳

Future Technology Roadmap

- Machine learning model refinement
- Deep learning integration
- Real-world testing and validation
- Continuous performance improvement

Challenges and Opportunities

- Complex urban infrastructure
- Diverse driving behaviors
- Technological innovation
- Safety and reliability focus