Problem Statement :

Self Driving Car in Virtual Environment using CNN and YOLOv8

Team Members
Isha Ghorpade
Vaishanavi Khade

Objectives

1. To simulate a self-driving car's visual system using deep learning.
2. To predict steering angles from road images using a CNN model.
3. To enhance scene understanding by integrating YOLOv8 for object detection.
4. To visualize the car's driving environment using OpenCV and Pygame.
5. To build a full AI pipeline from dataset generation to inference.

Benefits and Impact

1. Real-world applicability in autonomous vehicle systems.
2. Combines perception (YOLO) and control (CNN) modules.
3. Can be extended with lane detection, LSTM, or real-time control.
4. Helps understand integration of vision-based DL models.

Idea Details :

1. The project simulates a self-driving car's decision-making using a computer screen instead of real-world sensors.
2. Captures road frames using (ImageGrab) along with simulated steering, throttle, and speed values.
3. Trains a CNN (based on NVIDIA architecture) to predict steering angle.
4. YOLOv8 is used in parallel to detect pedestrians, vehicles, traffic signs, etc.Outputs are visualized using OpenCV and Pygame.

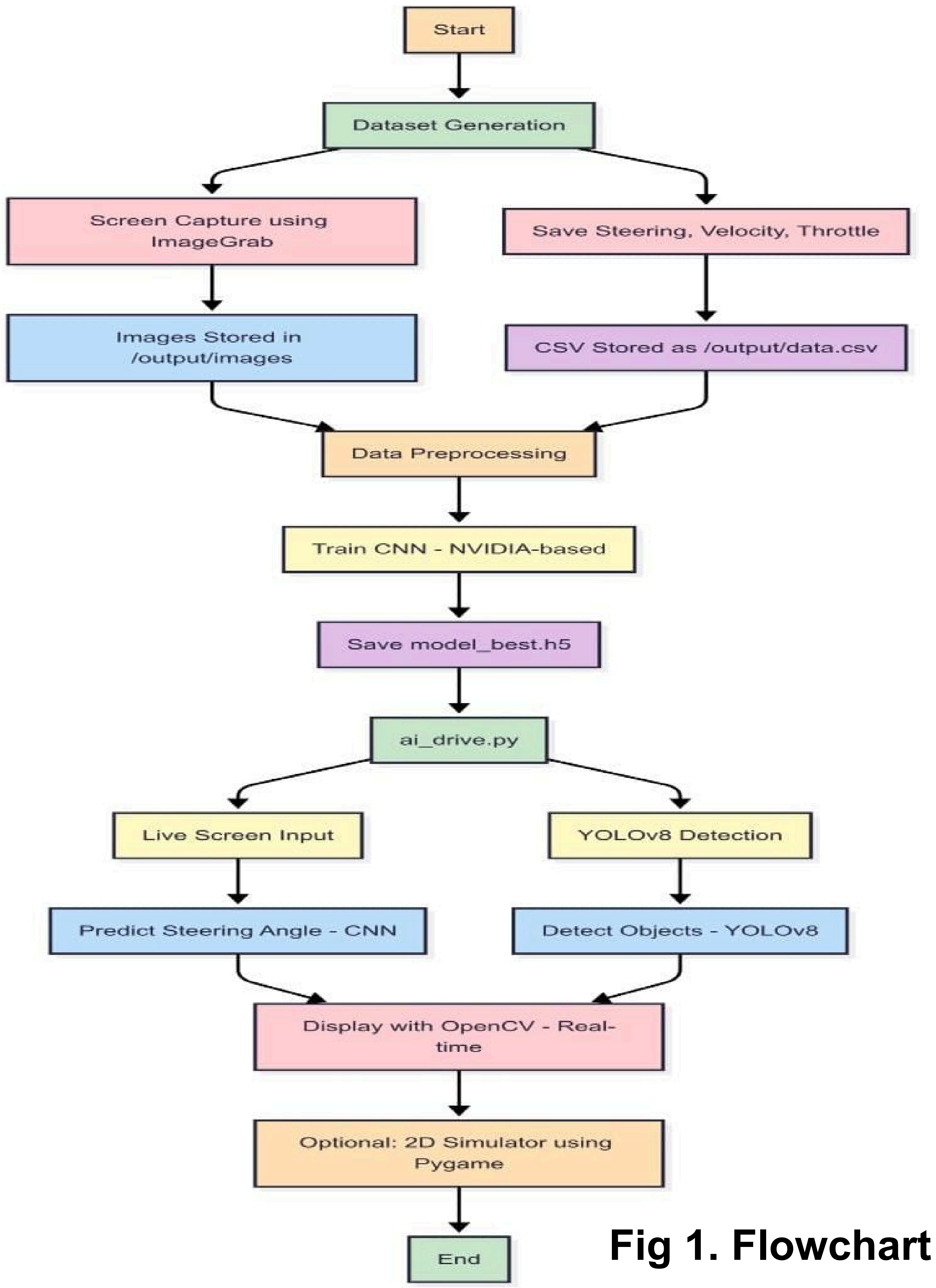


Fig 1. Flowchart

Literature Survey:

Survey on Methodologies

- Reviewed NVIDIA's end-to-end learning approach for self-driving.
- Studied YOLOv5/YOLOv8 for fast object detection.
- Compared CNN regression with traditional lane-based methods.

Survey on Gaps

- Most models use only steering prediction without object context.
- Few open datasets for simultaneous perception + control.
- Real-time simulator-free integration is rare.

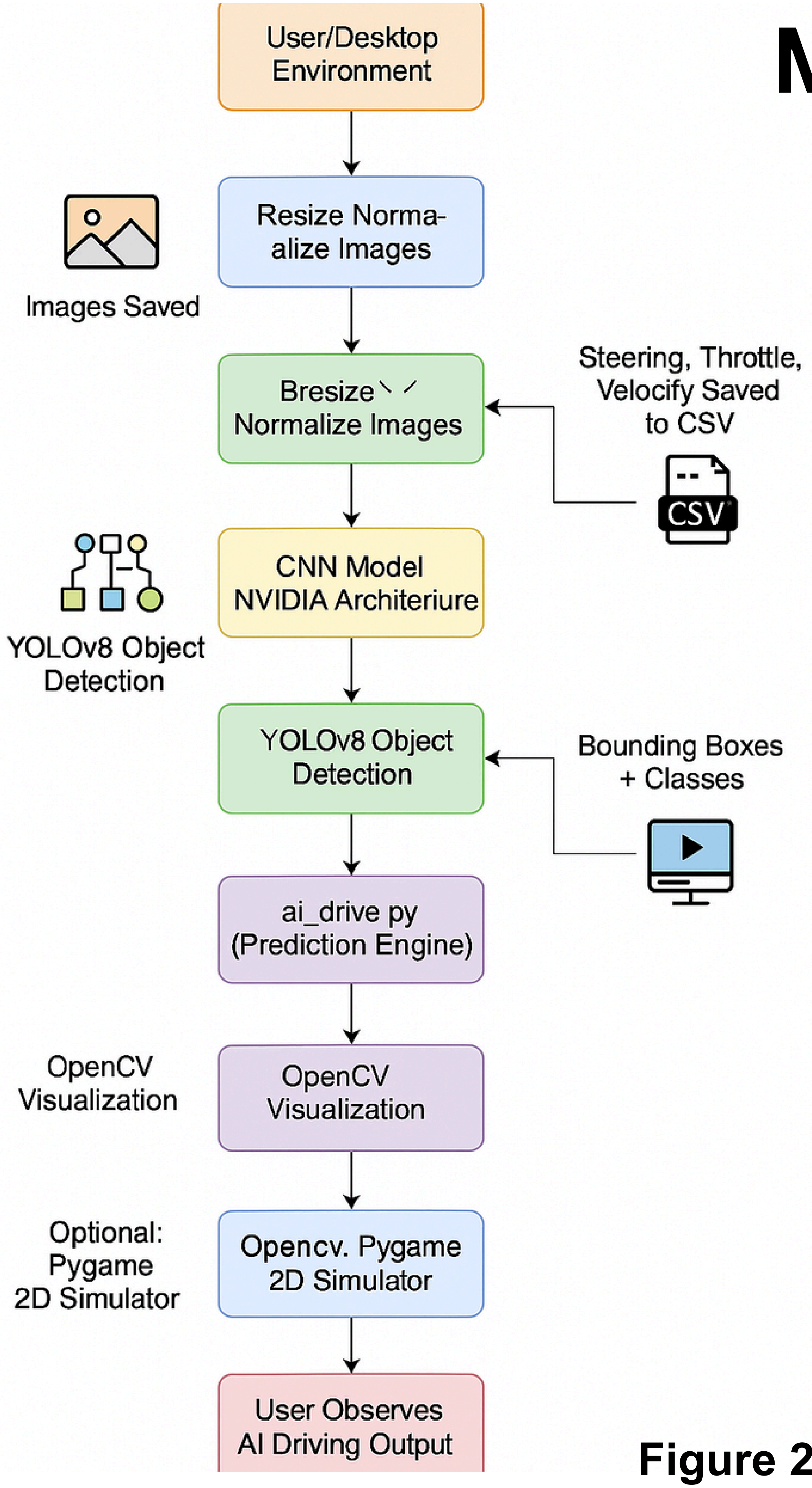
Survey on Datasets

- We generated our own dataset using screen capture + logging steering data in CSV.

Survey on results

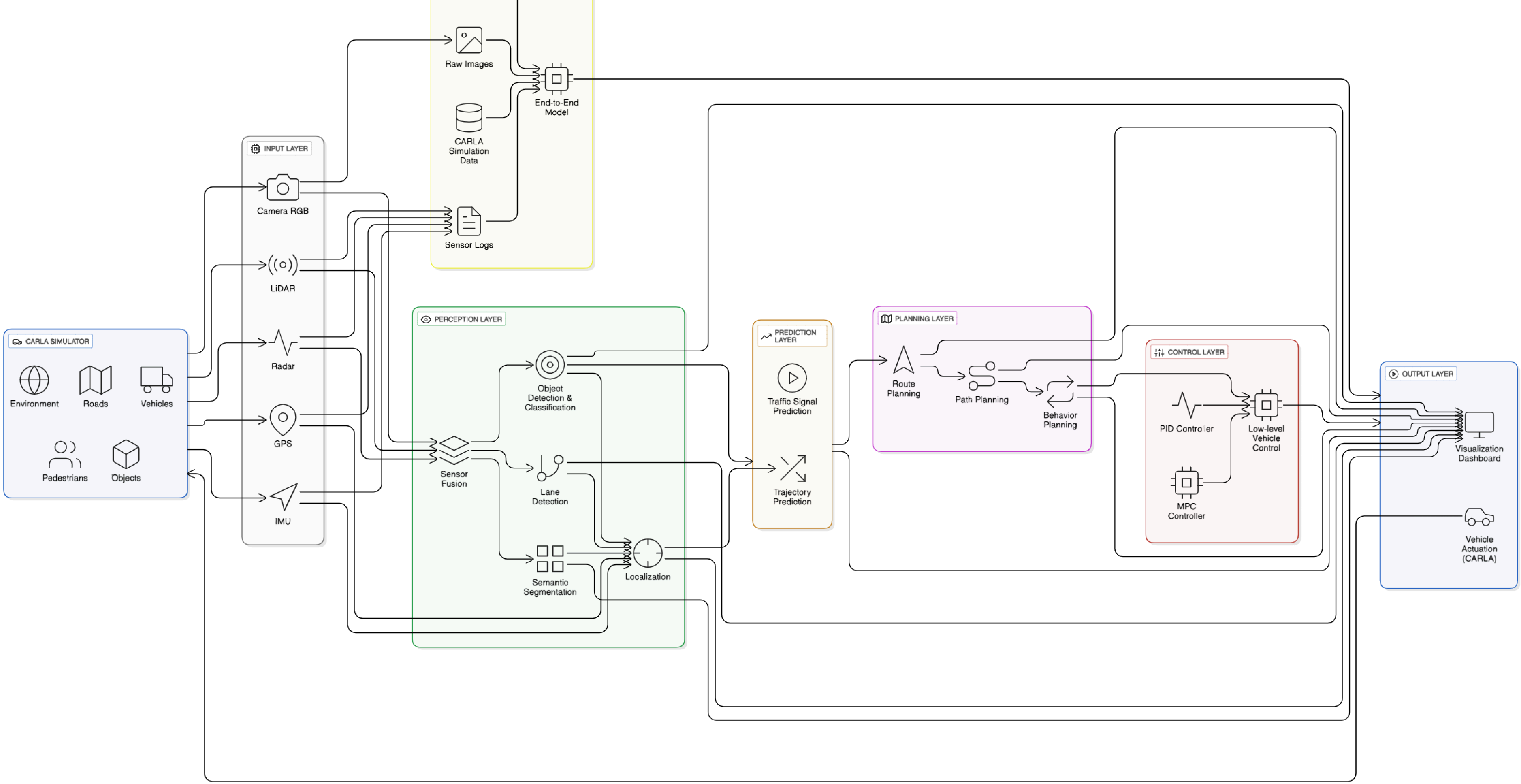
- End-to-end learning is effective for road following.
- Object detection greatly improves awareness and safety.
- Combining both in a lightweight pipeline is beneficial.

Methodology



- Image Collection & Preprocessing – Raw images are captured from the user/desktop environment, resized, and normalized to ensure uniform input quality. Simultaneously, steering, throttle, and velocity values are stored in a CSV file for training reference.
- CNN Model Integration – A Convolutional Neural Network (CNN) architecture is employed to process the normalized images and learn driving-related patterns.
- YOLOv8 Object Detection – The YOLOv8 model is applied to detect objects in the environment, generating bounding boxes and class labels for each detected object.
- Prediction & Visualization – The prediction engine (ai_drive.py) uses the processed inputs to make driving decisions, which are visualized through OpenCV.
- Simulation & Output – Optionally, a PyGame 2D simulator can be used to mimic real driving scenarios, while the user observes AI driving outputs and performance.

Figure 2: Architecture Diagram



Raw Images
CARLA Simulation Data
Sensor Logs
End-to-End Model
INPUT LAYER
Camera RGB
LiDAR
Radar
GPS
IMU
CARLA SIMULATOR
Environment
Roads
Vehicles
Pedestrians
Objects
PERCEPTION LAYER
Object Detection & Classification
Sensor Fusion
Lane Detection
Semantic Segmentation
Localization
PREDICTION LAYER
Traffic Signal Prediction
Trajectory Prediction
PLANNING LAYER
Route Planning
Path Planning
Behavior Planning
CONTROL LAYER
PID Controller
Low-level Vehicle Control
MPC Controller
OUTPUT LAYER
Visualization Dashboard
Vehicle Actuation (CARLA)

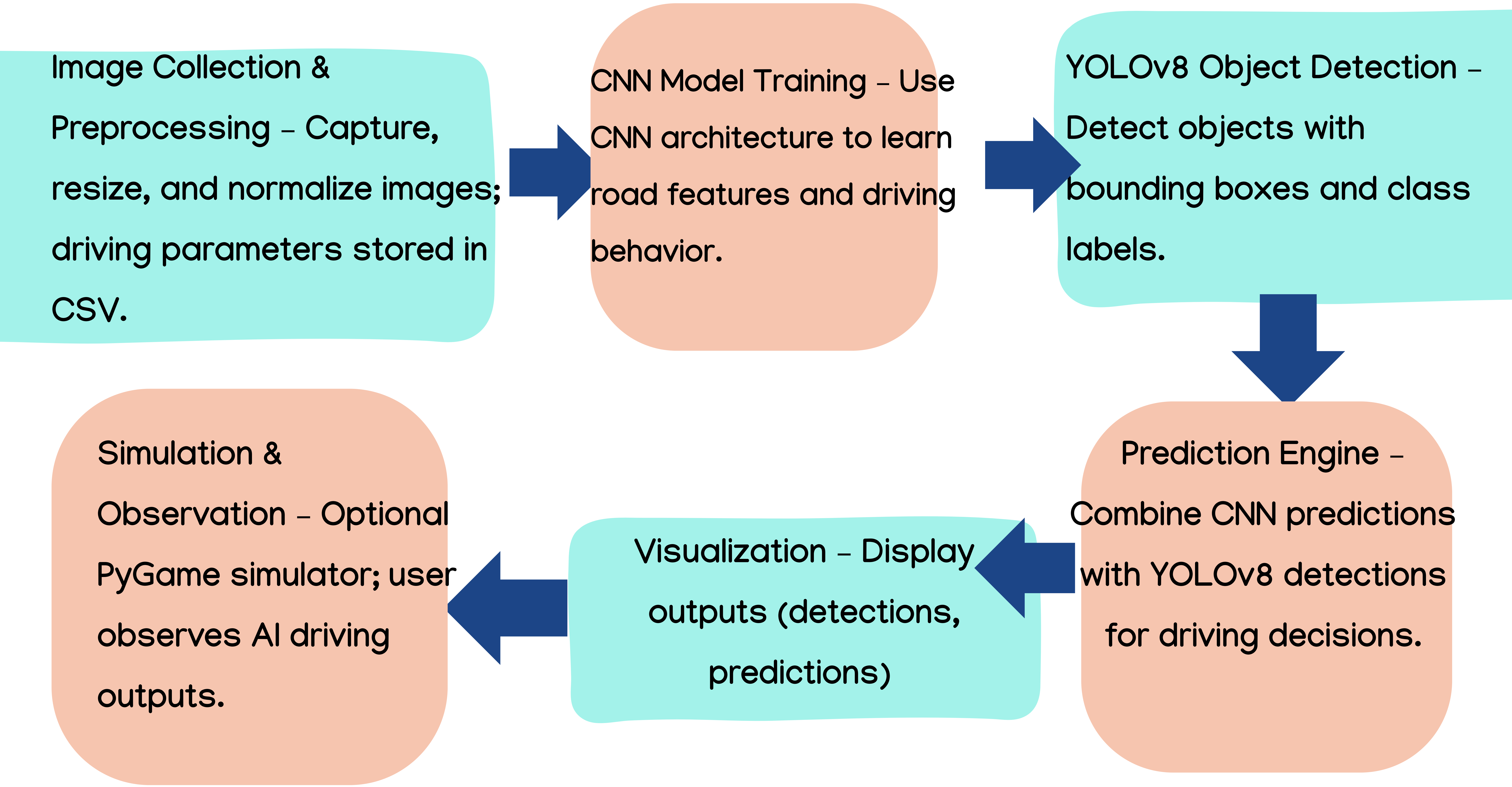
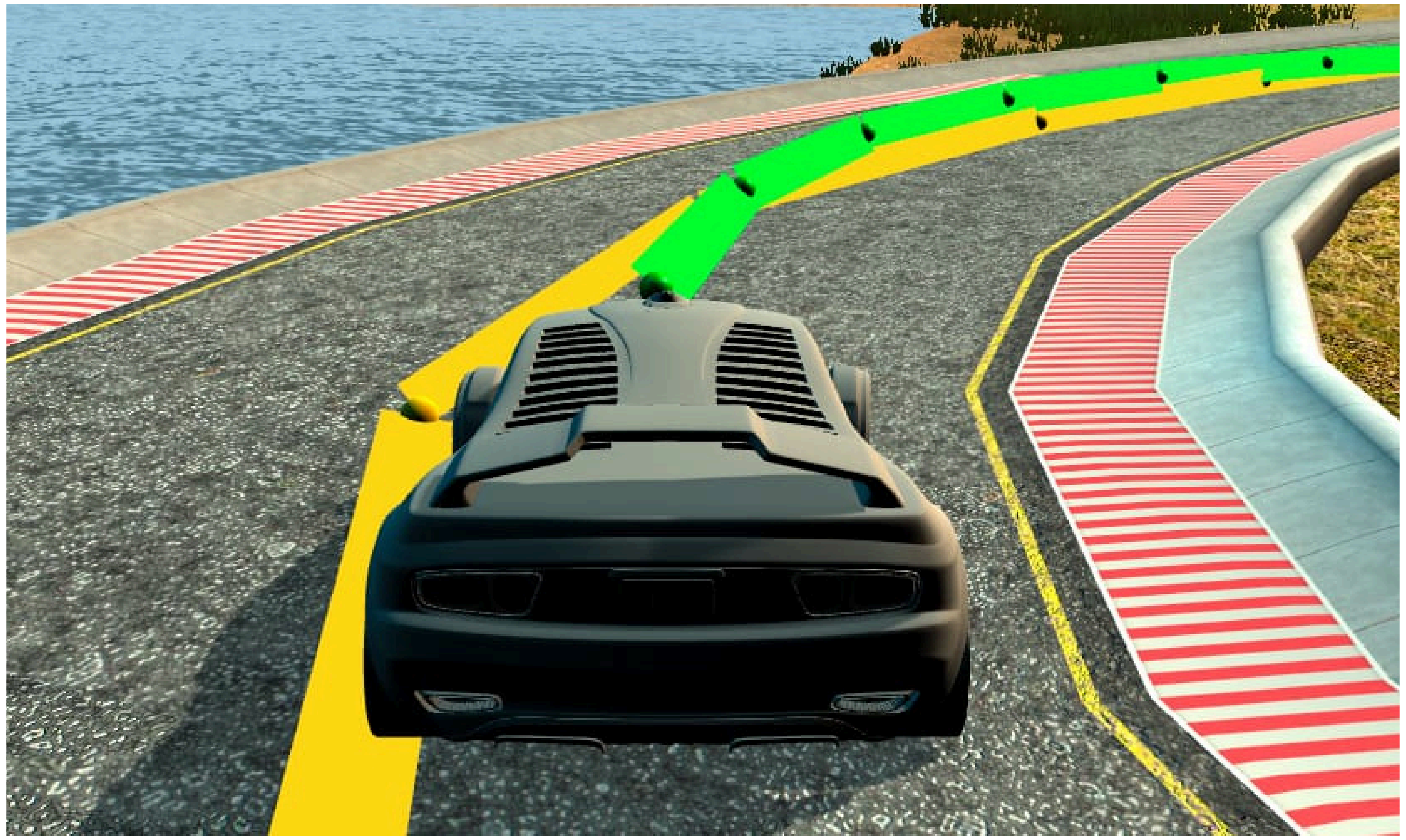


Image Collection & Preprocessing – Capture, resize, and normalize images; driving parameters stored in CSV.
CNN Model Training – Use CNN architecture to learn road features and driving behavior.
YOLOv8 Object Detection – Detect objects with bounding boxes and class labels.
Prediction Engine – Combine CNN predictions with YOLOv8 detections for driving decisions.
Visualization – Display outputs (detections, predictions)
Simulation & Observation – Optional PyGame simulator; user observes AI driving outputs.

Results



- CNN Only: Car drives autonomously on road, follows lane markings with predicted steering and throttle.
- CNN + YOLOv8: Adds real-time object detection with bounding boxes and class labels.
- Validation: CNN ensures lane following, YOLOv8 improves safety and awareness.

Right Curve Ahead
Curvature = 1357 m
Good Lane Keeping
Vehicle is 0.34 m away f
umbrella 0.54
car 0.86
Fremont Ave
Los Altos
3/4 MILE

- Object Detection – The output video successfully shows YOLOv8 detecting objects in the environment with bounding boxes and class labels.
- Prediction Alignment – CNN-based predictions (steering, throttle, velocity) are processed in sync with object detections.
- Visualization – OpenCV overlays bounding boxes and predictions directly onto video frames, providing clear real-time feedback.
- System Validation – The video confirms that the methodology flow (preprocessing → detection → prediction → visualization) works as designed.
- Performance Insight – Detected objects and predicted outputs demonstrate the model's capability to perceive surroundings and make driving-related decisions.

Tech Stack :

1. Language: PythonDeep
2. Deep Learning: TensorFlow, Keras
   → Custom CNN model based on NVIDIA end-to-end architecture used for steering angle prediction
3. Detection: YOLOv8 (Ultralytics)
4. Computer Vision: OpenCVScreen Capture: PIL.ImageGrabData
5. Logging: CSV, PandasOptional Simulation: PygameModel
6. Environment: Windows, VS Code

Innovativeness

1. Integrated two different deep learning models: CNN (for control) + YOLO (for perception).
2. Simulates self-driving visually without external hardware.
3. Fully modular system: data collection, training, and inference are decoupled.
4. Replaces physical sensors with screen emulation.

Conclusion

1. We will Built a full AI pipeline for vision-based steering control.
2. Successfully training and visualizing model predictions.
3. Adding object detection to improve environmental understanding.
4. **Future scope:** Use real video feeds, integrate lane detection, add vehicle control APIs.

THANK

YOU