AutoMIND
Autonomous Multimodal Intelligent Navigation Dashboard

Riccardo De Sanctis 1937859
Matteo De Sanctis 1937858

November 2, 2025

Abstract

We present a multimodal interactive cockpit prototype for an autonomous vehicle simulated with Webots and coordinated with SUMO. The dashboard aggregates live sensor streams (cameras, depth, radar, GPS, compass, accelerometer, speed), provides multimodal control (text, voice, and map-click) and demonstrates an extensible platform for prototyping human–vehicle multimodal interaction. This report summarizes design, implementation, key results, limitations, and future directions.

1 Motivation

The project is motivated by the rapid adoption of autonomous vehicles and the need for safe, usable interaction between humans and automated systems. Key drivers:

- Rise of autonomous vehicles: strong research and industry interest in safe autonomy.
- Safer and more intelligent navigation systems: robust sensor fusion and interaction improve safety.
- Enhance driver awareness and situational understanding by aggregating sensor streams in real time.
- Improve usability and reduce cognitive load via multimodal interfaces (speech, text, and direct map interaction).
- Bridge the gap between human intuition and autonomous decision-making (explainability & control fallback).
- Contribute to research in human–robot interaction and multimodal systems.

2 System architecture (high level)

The system consists of three main components:

Webots simulation - Physical vehicle model, world (village scenario), and sensors (cameras, rangefinder, radar, GPS, compass, accelerometer, speed).

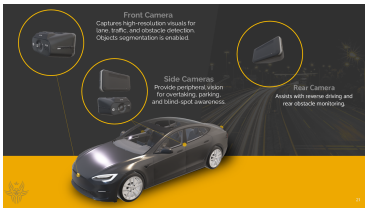
SUMO integration - SUMO runs traffic and handles vehicle low-level navigation. High-level destination requests (edge id + offset) are communicated from the dashboard/controller by writing a small file that the SUMO supervisor reads.

PyQt Dashboard - A Python GUI (PyQt5) that displays live sensor data, accepts multimodal inputs (text, voice, map clicks), and writes navigation requests to SUMO.

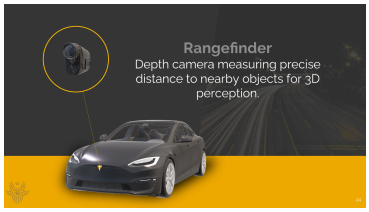
3 Sensors and UI elements

Each sensor is exposed in the dashboard with a compact visualization and short purpose. (An arrow from the car model in Fig.1 points to each device.)

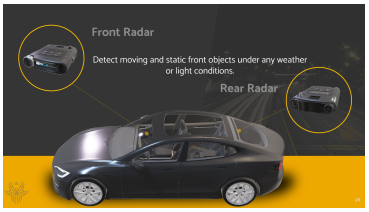
- **Front camera**: high-resolution frontal view for lane, traffic and obstacle detection; used for visual confirmation and potential object segmentation overlays.
- **Side cameras**: peripheral views for overtaking, parking and blind-spot monitoring.
- **Rear camera**: reverse driving and rear-obstacle visibility.
- **Rangefinder (depth)**: depth map / distance image enabling 3D perception and near-field distance measurement.
- **Radar (front & rear)**: detects moving and static objects across poor lighting / weather conditions; merged front+back radars yield near-360° coverage.
- **GPS / Mini-map**: vehicle geo-positioning for map visualization and click-based destination selection.
- **Compass**: heading information (north reference) for orientation and map alignment.
- **Speedometer**: current velocity gauge; needle-style speedometer plus a 180s history plot.
- **Accelerometer**: measures longitudinal acceleration/braking — used to drive throttle/brake bars in the UI.



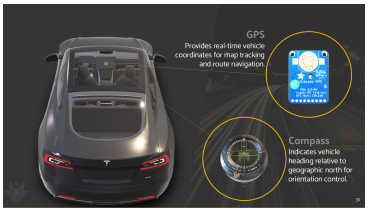
(a) Cameras.



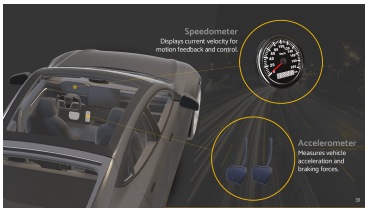
(b) Rangefinder.



(c) Radar.



(d) GPS and compass.



(e) Speed and acceleration.

Figure 1: Vehicle sensors used for perception, localization, and motion estimation.

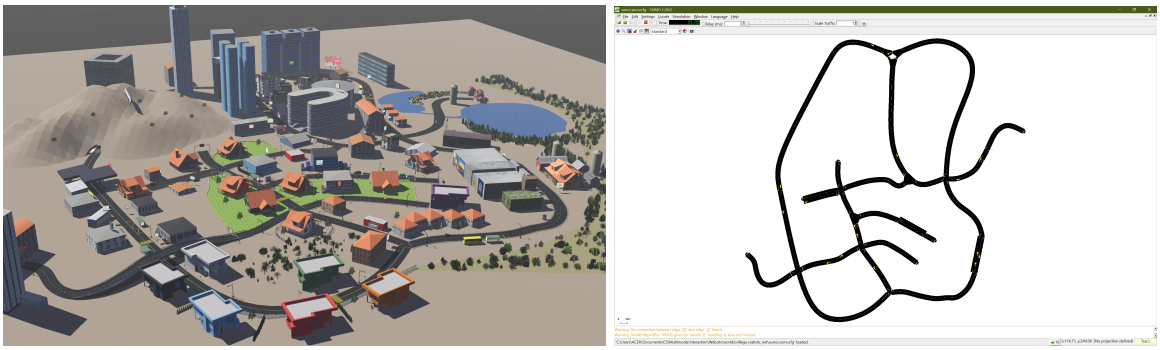
4 Implementation highlights

4.1 Webots and sensors

The vehicle controller enables sensors each timestep (camera.enable, gps.enable, radar.enable, etc.) and exposes raw arrays to the dashboard. Radar targets are read as objects (distance, azimuth, receiver_power, speed) and rendered top-down; depth is converted to grayscale.

4.2 SUMO orchestration

A small file-based handshake is used: the dashboard writes `new_dest.txt` containing the landmark SUMO edge id and offset; a supervisor thread in SUMO watches this file and updates the vehicle route. There are 80 available landmarks into the provided world. This lightweight approach simplifies multi-process coordination in a single-machine simulation. Similarly, for sensors that rely on the physics controlled by SUMO (speed and acceleration), `speed.txt` and `acceleration.txt` files are written by SUMO to communicate with the dashboard.



(a) Webots world. (b) SUMO engine interface.

Figure 2: Simulation environments used for autonomous vehicle testing and control.

4.3 Dashboard (PyQt)

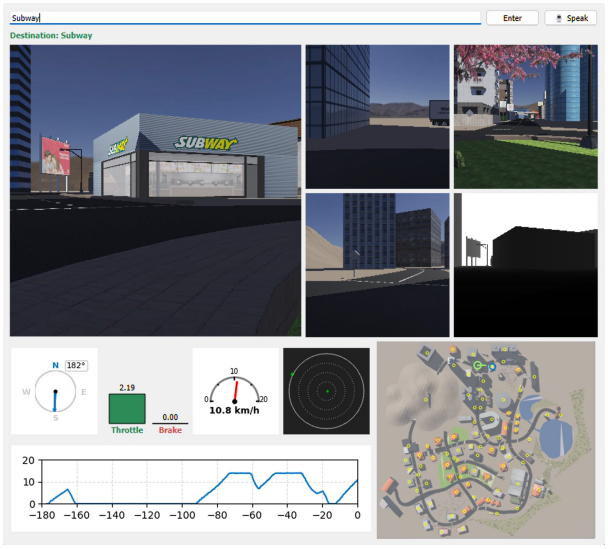


Figure 3: Dashboard accepting textual, vocal and map-click inputs. Front camera on the top left, side, rear and depth camera on the right top grid. Compass, acceleration and brake pedals, speedometer and speed history, and radar shown in the bottom left. The map shows landmarks (yellow circles), the vehicle location (green circle) and the selected location (blue circle).

Main features:

- Live camera widgets (front, left/right, rear, third person) using OpenCV → QImage conversion.
- Compass and speedometer implemented with matplotlib embedded canvases.

- Speed history uses a deque buffer and efficient line data updates to scroll the last 180 s.
- Mini-map: PNG top-down image with `world_to_pixel` / `pixel_to_world` conversion, click-to-select-nearest-landmark, and tooltip-on-hover for landmarks.
- Pedals widget: vertical throttle/brake bars driven by accelerometer sign and magnitude.
- Radar widget: combines front/back radar targets; azimuth rotated to match map orientation and points drawn with intensity tied to receiver power.
- Multimodal inputs: text box with fuzzy matching, speech recognition integration (SpeechRecognition + Google API or configurable local ASR), and direct map clicks.

5 Calibration, intermediate steps and practical engineering

Between prototyping milestones we did the following practical work that improved fidelity and usability:

- **Map calibration**: tuned `world_bounds` and small pixel offsets to align PNG map with Webots GPS coordinates.
- **Radar orientation**: tested azimuth offsets and combined front/back radars for full coverage.
- **ASR & translation**: integrated SpeechRecognition and optional translation (googletrans) for Italian → English preprocessing; resolved environment issues (Webots controller Python vs conda env).
- **Performance smoothing**: used a separate thread for Webots sensor polling and QTimer for GUI updates; adjusted timer to 20–33 ms for smooth visuals while avoiding CPU spikes.
- **Robustness**: handled missing devices, ensured clamping of radar ranges, and safeguarded file I/O between dashboard and SUMO.
- **UX polish**: added keyboard shortcuts (Enter / Insert), tooltips, and a small calibration UI for landmarks.

6 Demonstration & evaluation

We validated the integration qualitatively:

- Text/voice/map-click commands correctly selected landmarks and updated SUMO destination.
- Camera, depth, compass, radar and telemetry streamed to the dashboard at interactive rates (frame/timer dependent).
- Mini-map click produced expected pixel → world mapping after calibration; small residual offsets were corrected by tuning bounds.

Figure 4: Dashboard integrated in the vehicle.

Limitations: ASR quality depends on the recognizer; perfect object detection would require an onboard detector (YOLO/etc.) with bounding boxes synchronized to camera frames; the file-based SUMO handshake is simple but could be replaced by a messaging layer for scale.

7 Future work

Concrete extensions and research directions:

- Replace cloud ASR with local multilingual models (e.g., Whisper) and add intent parsing / NLU for richer commands.
- Add object detection and tracker (camera + radar + depth fusion) to display labeled bounding boxes and persistent tracks. Webots provide built-in recognition capabilities for cameras to perform object detection and segmentation; in a real world scenario, alternative approaches must be considered.
- Gesture and gaze control as additional modalities (camera-based hand/facial tracking).
- Adaptive UI: simplify information or surface alerts based on driver workload, preferences or context (task-aware interfaces).
- Augmented reality overlays on camera streams for navigation cues and warning highlights.
- Move to a robust IPC (ZeroMQ / gRPC / ROS topics) between dashboard, Webots, and SUMO; port to in-vehicle sources (CAN/ROS).
- Collaborative driving: share selected sensor summaries between vehicles for cooperative perception.

8 Conclusion

We implemented a compact, extensible multimodal cockpit that unifies simulation (Webots), traffic control (SUMO), and a PyQt dashboard supporting text, voice and map-based inputs. The platform demonstrates how multimodal interaction can improve situational awareness while remaining a flexible testbed for research (ASR, sensor fusion, adaptive interfaces).