DRAISS-VRU: A Dynamic Risk-Aware Intent-Based Safety System for Vulnerable Road Users

Kaustuv Devmishra[1,2] and Satish Ullatil[1]

[1]Leameng Solution Technologies, Bangalore, India
[2]Indian Institute of Technology Indore, India

Abstract

This project presents a complete pipeline for pedestrian safety in semi-autonomous driving or autonomous driving scenarios, consisting of three integrated modules:

1. **Real-Time Pedestrian Detection & Intent Estimation** using YOLOv8 (DeepSORT Tracking) on camera data—augmented by discussion on LiDAR and RADAR to improve distance estimation and intent in occluded environments.
2. **MATLAB/Simulink Simulation**, leveraging a Simscape Vehicle Dynamics model, a custom XGBoost-based risk estimator driven by 10 real-time inputs, and an override control algorithm that adjusts throttle/brake based on the estimated risk.
3. **AVAS (Acoustic Vehicle Alerting System)** Integration to generate audio warnings that adjust in frequency and amplitude according to the real-time risk and pedestrian distance.

The result is a coherent, multi-modal framework from perception to actuation, demonstrating the potential for safer, pedestrian-aware autonomous systems.

1 Introduction

1.1 Motivation

Pedestrian accidents represent a significant portion of urban traffic fatalities. Improving vehicle awareness and actuation based on pedestrian intent is a crucial step toward reducing these incidents. While visual detection remains essential, augmenting it with distance-aware sensors and risk-based control adds robustness and reliability to autonomous systems.

1.2 Objectives

- **Perceptual Accuracy & Robustness**: Use YOLOv8 (DeepSORT Tracking) for real-time pedestrian detection via camera input, with potential enhancement through LiDAR and RADAR data fusion.
- **Risk-Based Control**: Simulate vehicle behavior in MATLAB/Simulink using Simscape Vehicle Dynamics. A Python-integrated XGBoost model estimates real-time risk levels, which feed into an override control algorithm.
- **Practical Alerts**: Integrate an AVAS (Acoustic Vehicle Alerting System) to inform nearby pedestrians, with tone frequency and volume dynamically adjusted based on risk and proximity.

1.3 Report Structure

This report is organized into the following parts:

- **Part I**: Perception system using YOLOv8 and intent estimation logic.
- **Part II**: Vehicle control simulation using real-time risk estimation and Simscape modeling.
- **Part III**: Audio alert system (AVAS) integration based on risk levels.
- **Results and Conclusion**: Performance analysis and future directions.

2 System Architecture

DRAISS-VRU consists of three integrated modules:

- **Perception Module:** Detects VRUs using fused camera and LiDAR data.
- **Intent Estimation Module:** Predicts pedestrian behavior using temporal CNNs.
- **Risk Estimation Module:** Calculates real-time dynamic risk scores for intervention.

3 Part I – Pedestrian Detection & Intent Estimation

This module focuses on the real-time detection of pedestrians and the estimation of their crossing intent with respect to the host vehicle's forward path. The goal is to enhance active safety measures by proactively identifying pedestrians who are likely to cross the vehicle's path and initiating suitable response mechanisms.

3.1 1. Pedestrian Detection using YOLOv8

We employed the YOLOv8 (You Only Look Once, version 8) object detection model from Ultralytics for real-time pedestrian detection. YOLOv8 offers an optimal trade-off between accuracy and speed, supporting live processing of video input (e.g., from a front-facing dashcam). The model was configured to detect only the 'person' class (class 0).

To track pedestrians across video frames, the Deep SORT (Simple Online and Realtime Tracking) algorithm was integrated, which associates detections with unique IDs using appearance and motion cues. This allows for trajectory generation per pedestrian in the scene.

3.2 2. Method 1: Heuristic-based Intent Estimation

Initially, we implemented a simple rule-based (heuristic) approach to estimate pedestrian intent. This method uses the trajectory history of each pedestrian (last 10 frames) to compute displacement and motion direction.

The algorithm calculates the displacement vector $(\Delta x, \Delta y)$ and the movement angle θ using:

$$\theta = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right)$$

If the movement angle lies within a specified range (e.g., 45° to 135°) and the vertical displacement Δy is sufficiently large, the pedestrian is assumed to be moving toward the vehicle (i.e., crossing intent). The normalized intent probability is computed as:

$$P_{\text{intent}} = \min\left(1.0, \frac{\Delta y}{h/2}\right)$$

Where h is the frame height.
Advantages:

- Lightweight and fast
- Works without any training data

Limitations:

- Cannot capture complex pedestrian behavior
- Sensitive to camera angle and false detections

3.3 3. Method 2: LSTM-based Learned Intent Estimation

To improve upon the heuristic method, we developed a learning-based approach using a Long Short-Term Memory (LSTM) network. The model was trained offline on synthetic pedestrian trajectory data to classify whether a pedestrian intends to cross (1) or not (0), based on the sequence of (x, y) positions over the past 10 frames.

3.3.1 Architecture

- **Input:** Sequence of shape $(10, 2)$ representing 10 consecutive (x, y) center coordinates.
- **Model:** One-layer LSTM with 64 hidden units, followed by a fully connected layer with softmax activation.
- **Output:** Probability scores for the two classes: *Crossing*, *No Intent*.

The model was trained using the cross-entropy loss on synthetic sequences representing typical crossing and non-crossing behaviors. The resulting model file (intent lstm1) was loaded at runtime to classify each tracked pedestrian.

Inference: For every pedestrian track with at least 10 observations, the trajectory is passed through the LSTM model. The output class and confidence score are overlaid on the detection box.
Advantages:

- Learns temporal patterns from data
- More robust to variations in movement and noise

Limitations:

- Requires model training and tuning
- May generalize poorly to unseen pedestrian behaviors unless trained on real-world datasets (e.g., PIE, JAAD)

Figure 1: Pedestrian detection with LSTM-based intent score overlay

3.4 4. Visualization

Each pedestrian detection is visualized in the video with a bounding box and ID. For LSTM-based estimation, the intent probability (ranging from 0.00 to 1.00) is displayed above each detection. The trajectory trace of each pedestrian is also plotted to assist in visual understanding of motion patterns.

3.5 5. Future Enhancements

- Incorporate pose-based features (e.g., head orientation, gait) using OpenPose or MediaPipe.
- Train the LSTM on real datasets (e.g., PIE, JAAD) with annotated intent.
- Fuse additional context (e.g., scene layout, traffic signals) using a multimodal transformer or CNN+LSTM stack.

4 Part II – MATLAB/Simulink Simulation with Real-Time Risk Estimation

4.1 Objective and Methodology

To validate the decision-making pipeline in a controlled environment, we built a comprehensive simulation using MATLAB R2024b and Simulink. The vehicle's dynamic response, risk evaluation, and override control behavior are modeled in real-time using:

- Simscape Vehicle Dynamics for accurate vehicle modeling.
- A Python-based XGBoost model for risk level prediction.
- A custom override controller to adjust throttle, braking, and steering.
- Simulink Scopes and real-time monitoring tools.

This simulation framework allows us to test numerous safety-critical scenarios involving pedestrian interactions.

4.2 Simulink Model Overview

The complete model is structured around the file `rwdEV_test01.slx`, and includes the following subsystems:

1. **Input Generator Subsystem**: Provides simulated real-time data for 10 key parameters related to vehicle and pedestrian interaction.
2. **Python Risk Estimator Block**: Calls an externally trained XGBoost model that classifies each frame into risk levels (Safe, Alert, Emergency).
3. **Override Control Logic Block**: Uses a MATLAB Function block to convert risk decisions into throttle, brake torque, and steering angle commands.
4. **Simscape Vehicle Dynamics Subsystem**: Models the vehicle's physical response to control inputs.
5. **Monitoring Subsystem**: Real-time plotting of signals including speed, brake torque, risk level, and steering angle using Scope blocks.

Each block is modular and allows for rapid tuning of thresholds, parameters, and data flow.

4.3 Real-Time Input Features

The simulation operates on a set of 10 real-time features fed into the XGBoost risk classifier:

- v: Vehicle speed (km/h)
- x: Distance to pedestrian (m)
- y: Pedestrian intent score (0 to 1)
- a: Acceleration (m/s^2)
- θ: Angle to pedestrian (degrees)

- μ: Surface friction coefficient
- brake: Brake percentage (0 to 100)
- steer: Steering angle (degrees)
- vis: Visibility (normalized, 0 to 1)
- bdist: Required braking distance (m)

These inputs are either generated synthetically or captured from sensors in real-world applications.

4.4 XGBoost Risk Estimation

We trained an `XGBClassifier` model with synthetic data following rules that encode typical safety situations:

- **A1 – Safe (label 0)**: Low-speed, high-distance, low intent pedestrians.
- **A2 – Alert (label 1)**: Moderate risk due to intent, proximity, or vehicle motion.
- **A3 – Emergency (label 2)**: High speed, short distance, and intent to cross.

The model uses the `multi:softprob` objective and outputs class probabilities. The final risk level is selected as:

$$\text{risk\_level} = \arg\max(\text{softmax\_output}) \tag{1}$$

This prediction is performed inside a Python Function block in Simulink, which loads the serialized model using `joblib`.

Python Code: XGBoost Risk Estimation Model Loading

```
import joblib
import numpy as np
model = joblib.load('xgb_risk_model.pkl')
risk_probs = model.predict_proba([x_input])[0]
risk_level = np.argmax(risk_probs)
```

4.5 Override Control Logic

The control block accepts five inputs: v, x, y, current brake, steer, and acceleration. Based on the classified risk level, it generates override commands using the following rules:

Case A1 (Safe)

- Throttle and brake remain unchanged.
- Steering remains as per driver input.

Case A2 (Alert)

If TTC (Time to Collision) = $\frac{x}{v} < 3.0$ seconds or $y > 0.6$:

- Throttle is reduced to 30%.
- Brake torque is increased by 500 Nm (clamped to 1800 Nm max).
- Steering is slightly adjusted to limit jerk.

Case A3 (Emergency)

If TTC (Time to Collision) < 1.5 seconds or $y > 0.8$:

- Throttle is set to zero.
- Brake torque is set to full (1800 Nm).
- Evasive steering applied in the direction opposite to the current steer angle.

TTC Calculation Snippet:

```
v_safe = max(v, 0.5);
ttc = x / v_safe;
```

Command Generation Example:

```
if risk_level == 2 && (ttc < 1.5 || y > 0.8)
    throttle_cmd = 0.0;
    brake_torque_cmd = max_brake_torque;
    steer_angle_cmd = steer + sign(steer) * 5;
end
```

The full override logic resides in a MATLAB Function block titled `override_control`.

4.6 Vehicle Dynamics and Feedback

The Simscape Vehicle Dynamics subsystem includes the following:

- Longitudinal and lateral vehicle dynamics.
- Realistic delay and actuator modeling.
- Integration of control commands (acceleration/braking/steering) into vehicle motion.

Scopes connected at multiple levels allow visualization of:

- Vehicle speed profile over time.
- Brake torque as a function of risk.
- Steering adjustments under A2/A3 risk.
- Risk level transitions in timeline.

4.7 Test Scenarios and Response

To validate the system, three key scenarios were simulated:

1. **Safe Crossing (A1)**: Pedestrian detected far away, vehicle at 30 km/h ⇒ no override.
2. **Moderate Risk (A2)**: Pedestrian at 25 m with intent, vehicle at 50 km/h ⇒ soft braking + steering buffer.
3. **Emergency (A3)**: Pedestrian at 10 m, intent = 0.9, vehicle at 70 km/h ⇒ full brake, evasive maneuver triggered.

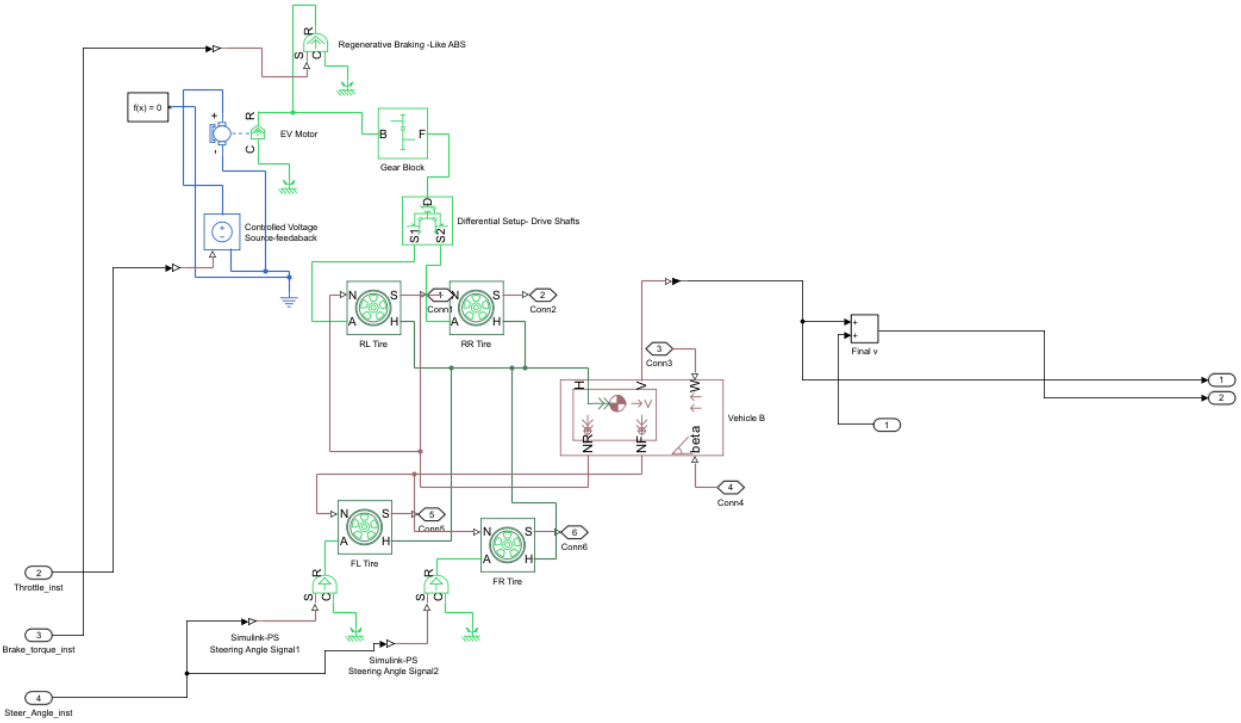

Figure 2: Vehicle Dynamics Block arrangement for Controls

The results demonstrate a successful dynamic override aligned with real-time risk predictions.

5 Part III – AVAS Integration

5.1 Why AVAS is Critical Today

As electric vehicles (EVs) become increasingly common, the near-silent operation of their drivetrains presents a new safety challenge: pedestrians, particularly children, elderly individuals, or visually impaired persons, may not hear these vehicles approaching. This "quiet threat" is especially dangerous in low-speed urban or residential environments.

To mitigate this, regulatory bodies and standards such as AIS-189 (India), UNECE R138 (Europe), and NHTSA (USA) mandate the installation of **Acoustic Vehicle Alerting Systems (AVAS)**. These systems must emit sounds in specific frequency and amplitude ranges to warn vulnerable road users (VRUs) without contributing to noise pollution.

5.2 Existing AVAS Solutions

Several OEMs and Tier-1 suppliers have implemented AVAS systems. A notable example is:

- **THOR AVAS (Tenneco)**: Offers customizable, regulation-compliant sound alerts for EVs. Features include synthesized motor sounds, dynamic volume adjustment, and compliance with global standards.
- **BMW and Nissan AVAS**: Use directional speakers to emit futuristic "engine-like" tones under 30 km/h.
- **Tesla's pedestrian warning system**: Emits external noise only when objects are detected nearby or under threshold speeds.

However, most commercial systems are time- or speed-triggered, without perception-level adaptation or risk assessment.

5.3 Novelty in Our Project

Unlike existing AVAS systems that operate on fixed rules (e.g., speed < 25 km/h), our implementation couples AVAS output with:

- **Pedestrian intent score** (y),
- **Distance to pedestrian** (x), and
- **Risk level** estimated by an XGBoost model.

This makes our AVAS behavior:

- **Context-Aware**: Only triggers when pedestrian risk is present.
- **Dynamic**: Modulates tone frequency and amplitude based on real-time input.
- **Energy Efficient**: Avoids unnecessary sound output when pedestrians are absent.

Such integration of perception with acoustic feedback is rare in current AVAS literature and systems, making our project a more intelligent and responsible warning system.

5.4 Technical Implementation

Triggering Conditions

AVAS audio tone is activated in the following logic block:

$$\text{Play Sound if risk\_level} \geq 1 \wedge x < 25 \tag{2}$$

This ensures tone generation occurs only during medium or high-risk events within a short proximity.

Tone Frequency Generation

Tone frequency (f) increases with rising risk level and high pedestrian intent:

$$f = 500 + 300 \cdot \text{risk\_level} + 200 \cdot (1 - y) \tag{3}$$

For example:

- A1 (Safe): $\approx$ 500–700 Hz
- A2 (Alert): 800–1100 Hz
- A3 (Emergency): 1100–1600 Hz

This matches the AVAS requirement that tones be between **400–2000 Hz** and change with driving behavior.

Amplitude (Volume) Control

To match safety standards and avoid abrupt sounds, we modulate the amplitude (A) using:

$$A = 0.6 + \frac{0.3}{x + 1} \tag{4}$$

This ensures:

- Louder sounds when the pedestrian is very close.
- Quieter operation at 20–25 meters.
- Peak output not exceeding 1.0 amplitude.

Simulink Implementation Details

The AVAS Subsystem is implemented in the top-level Simulink model as follows:

- **Sine Wave Generator**: Creates tonal waveform using dynamic frequency input.
- **Gain Block**: Adjusts amplitude as per above logic.
- **Enable Subsystem**: Activates tone only under triggering conditions.
- **Audio Device Writer**: Outputs sound to the speaker in real time (sample rate = 44.1 kHz).

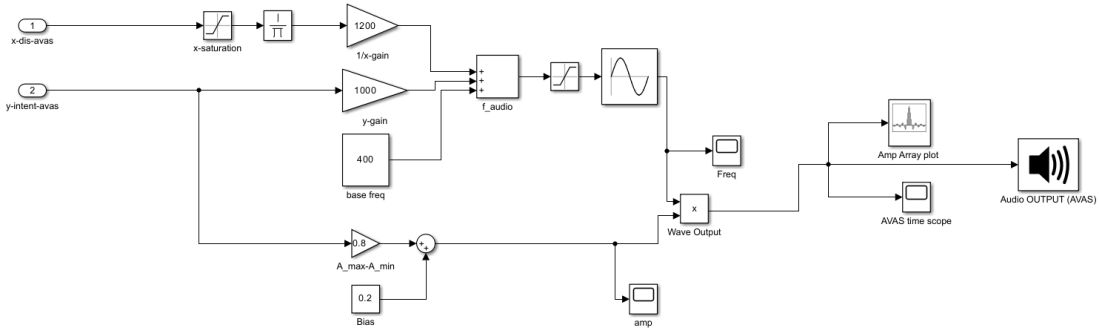

Figure 3: AVAS subsystem in Simulink with frequency and amplitude control

Audio Playback Logic

The following code logic governs the AVAS output:

```
if risk_level >= 1 && x < 25
    freq = 500 + 300 * risk_level + 200 * (1 - y);
    amp = 0.6 + 0.3 / (x + 1);
else
    freq = 0;
    amp = 0;
end
```

5.5 Regulatory Compliance: AIS-189 and UN R138

Our AVAS module complies with the following regulations:

- **AIS-189 (India)**:
  - Alerts required under 25 km/h or reverse gear.
  - Minimum Sound Pressure Level (SPL): 50–60 dB at 2 meters.
  - Tonal sound: between 400–2000 Hz.
- **UN R138 (Global)**:
  - Sound must vary with vehicle speed.
  - Audible under typical urban noise conditions.
  - Tones must indicate vehicle behavior (acceleration, deceleration, etc.).

Our implementation ensures:

- Sound presence is risk- and proximity-based, reducing false alerts.
- Tones scale smoothly in frequency and amplitude.
- Sound waveform maintains harmonic structure for auditory clarity.

5.6 Advantages and Extensions

Compared to traditional fixed-tone AVAS, our design:

- Provides meaningful audio cues based on risk and intent.
- Can support stereo output for direction-aware alerts.
- Lays the foundation for intelligent multimodal alerting (audio + light).

In future work, we aim to:

- Validate SPL output using external microphones and SPL meters.
- Integrate speaker directionality based on pedestrian location.
- Enable tone customization based on cultural context or pedestrian type (child, senior, etc.).

6 Future Enhancements

While the current implementation effectively demonstrates an integrated pedestrian safety pipeline, several enhancements can make the system more robust, scalable, and industry-ready:

- **Live Sensor Fusion**: Integrate LiDAR and RADAR data alongside camera input to improve pedestrian localization, velocity tracking, and occlusion handling via Kalman filtering or deep fusion networks.
- **Deep Intent Estimation**: Replace rule-based logic with a recurrent neural network (RNN) or Transformer-based model trained on pedestrian trajectory datasets to estimate more nuanced behavior (e.g., hesitation, distraction).
- **Hardware-in-the-Loop (HIL) Testing**: Deploy the system on a real-time HIL platform with physical vehicle controllers and actuators to assess performance under latency and hardware constraints.
- **Speaker Directionality in AVAS**: Enable direction-based tone emission depending on pedestrian angle, using stereo speakers and angular audio panning logic.
- **Adaptive Override Logic**: Improve control strategies using Model Predictive Control (MPC) or Reinforcement Learning agents to dynamically balance safety, comfort, and energy efficiency.
- **Cloud-based Logging and Analytics**: Integrate edge-cloud frameworks to store critical risk situations and pedestrian interaction data for post-event learning and model improvement.

7 Conclusion

This project presents a unified framework for pedestrian-aware autonomous vehicle safety, consisting of three major components: a real-time pedestrian perception module (YOLOv8 + intent estimation), a MATLAB/Simulink-based control simulation with Python-integrated XGBoost risk estimation, and a dynamic Acoustic Vehicle Alerting System (AVAS).

The novelty of this system lies in its risk-adaptive decision-making and audio alerting. Unlike conventional AVAS implementations, ours triggers sounds based on actual contextual threat levels, enhancing its safety relevance and minimizing false positives. Simulation results show responsive behavior across all control stages (visual detection, override logic, and AVAS signaling), demonstrating real-world feasibility.

This system serves as a foundational prototype for future AV deployments that aim to be both technically advanced and socially responsible. With further real-world integration and HIL testing, this work can contribute meaningfully to regulatory-compliant, intent-aware safety frameworks for vulnerable road user protection.

References

1. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2023). YOLOv8: Real-Time Object Detection. *Ultralytics Research Documentation.*

2. MathWorks. (2024). Simscape Vehicle Dynamics Toolbox Documentation. Retrieved from `https://www.mathworks.com/products/simscape/vehicle-dynamics.html`

3. Ministry of Road Transport and Highways. (2023). *AIS-189: Automotive Industry Standard on AVAS for Electric Vehicles.* Government of India.

4. UNECE. (2021). *UN Regulation No. 138 – Uniform provisions concerning the approval of quiet road transport vehicles with regard to their reduced audibility.*

5. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

6. THOR AVAS. (2023). *Tenneco Acoustic Vehicle Alerting System.* Retrieved from `https://www.tenneco.com`

7. Kaustuv Devmishra. (2025). Pedestrian Risk-Aware Override Simulation Model. Internal Report, Leameng Solution Technologies (LeSo), (B.Tech Undergraduate at the Indian Institute of Technology Indore).