Multi-Camera Vision System with Priority-Based Control for Autonomous Vehicle Guidance in Webots

1st Grigorev Ilya
DS-01
Innopolis University
Innopolis, Russia
il.grigorev@innopolis.university

2nd Vladimir Toporkov
RO-01
Innopolis University
Innopolis, Russia
v.toporkov@innopolis.university

3rd Salavat Faizullin
DS-01
Innopolis University
Innopolis, Russia
s.faizullin@innopolis.university

***Abstract*—This paper presents a novel priority-based multi-camera control system for autonomous vehicle lane detection in Webots simulation. The system employs three cameras with a sophisticated control strategy: the central camera maintains primary authority when detecting lane markings, while left and right cameras (angled at 45 degrees) provide fallback guidance. A unique control arbitration mechanism prevents conflicts between peripheral cameras by granting exclusive control to the camera with higher detected error when the central camera fails to detect lanes. The vision system is coupled with a PD controller [6], [7] that processes normalized error signals to generate precise steering commands. Experimental results demonstrate robust lane-following performance through this integrated vision-based control approach.**

***Index Terms*—autonomous vehicles, multi-camera systems, lane detection, control arbitration, PD controller, computer vision, Webots**

I. Introduction

Autonomous vehicle navigation requires reliable perception systems that can maintain performance under varying environmental conditions. Multi-camera configurations offer enhanced robustness through sensor redundancy and complementary field-of-view coverage. However, effective fusion of multiple visual inputs presents significant challenges in control arbitration and conflict resolution.

This paper presents a priority-based control system for autonomous lane following that utilizes three cameras with distinct roles and a deterministic arbitration strategy. The system employs classical computer vision techniques including Canny edge detection [1] and Hough Transform [2] for lane marking identification, combined with a novel control protocol that ensures unambiguous steering decisions. The vision subsystem is integrated with a PD controller that processes error signals to generate vehicle control commands [6].

II. System Architecture

A. Multi-Camera Configuration

The vision system comprises three cameras with specific orientations and responsibilities:

Github repository: https://github.com/Spectre113/HighV

- **Center Camera**: Forward-facing (0°) for primary lane tracking
- **Left Camera**: Angled 45° left for left-side lane detection
- **Right Camera**: Angled 45° right for right-side lane detection

Each camera independently processes images through identical computer vision pipelines: Region of Interest (ROI) extraction, HSV color segmentation for yellow lane detection, Canny edge detection [1], and probabilistic Hough Transform [2] for line segment identification.

B. Priority-Based Control Strategy

The core innovation of our system is the hierarchical control strategy:

Initialize: $active\_camera \leftarrow center$
while vehicle running **do**
  $center\_lines \leftarrow detectLines(center\_image)$
  $left\_lines \leftarrow detectLines(left\_image)$
  $right\_lines \leftarrow detectLines(right\_image)$
  if $center\_lines \geq 1$ **then**
    $active\_camera \leftarrow center$
  else if $active\_camera \neq center$ **then**
    // Maintain current side camera control
  else
    $left\_error \leftarrow calculateError(left\_lines)$
    $right\_error \leftarrow calculateError(right\_lines)$
    if $|left\_error| > |right\_error|$ **then**
      $active\_camera \leftarrow left$
    else
      $active\_camera \leftarrow right$
    end if
  end if
  $control\_error \leftarrow getError(active\_camera)$
  $steering \leftarrow PD\_controller(control\_error)$
end while

III. Technical Implementation

A. Region of Interest Optimization

To optimize computational efficiency and focus processing on relevant image regions, we implement camera-specific ROIs applied before color segmentation. The ROIs are defined in the format [y_start, y_end, x_start, x_end]:

- **Central Camera**: [height // 4, -1, 0, -1] - Processes the lower 75% of the image (from 1/4 height to bottom) across the full width, focusing on the road area immediately ahead of the vehicle.
- **Left Camera**: [height // 2, -1, width // 2, -1] - Processes the lower 50% of the image (from 1/2 height to bottom) and the right half of the image (from 1/2 width to right edge), capturing the right-side road area as the camera is angled left.
- **Right Camera**: [height // 2, -1, 0, width // 2] - Processes the lower 50% of the image (from 1/2 height to bottom) and the left half of the image (from left edge to 1/2 width), capturing the left-side road area as the camera is angled right.

This ROI optimization reduces computational load by 50-75% while maintaining relevant visual information for lane detection, significantly improving real-time performance.

B. Vision Processing Pipeline

Each camera follows an identical processing sequence after ROI extraction:

1. **Color Segmentation**: Conversion to HSV color space and thresholding for yellow lane markings:

$$mask = inRange(hsv, [20, 30, 30], [30, 255, 255]) \quad (1)$$

2. **Edge Detection**: Canny edge detection [1] on masked regions with thresholds (250, 252)

3. **Line Detection**: Probabilistic Hough Transform [2] with parameters:

- Threshold: 50 votes
- Minimum line length: 30 pixels
- Maximum line gap: 20 pixels

C. Error Calculation and Normalization

For each detected line segment, midpoints are calculated as:

$$mid_x = \frac{x_1 + x_2}{2}, \quad mid_y = \frac{y_1 + y_2}{2} \quad (2)$$

The control error is computed from the active camera's midpoints:

$$error = \frac{2 \times (\frac{\sum mid_x}{N} - \frac{image\_width}{2})}{image\_width} \quad (3)$$

This normalized error in the range [-1, 1] provides input to the PD controller for steering adjustment.

D. Lane Following using PD-controller

The control system employs a Proportional-Derivative (PD) controller [6], [7] as the primary control stage, which processes the normalized vision error to generate steering commands. During this initial control phase, the system simultaneously collects and stores GPS coordinates of the vehicle's path for subsequent analysis and optimization.

The PD controller operates with the following fundamental parameters [8]:

- **Speed**: 18 km/h (vehicle operating speed)
- **Sampling Time**: 32 ms (control loop period)
- **Proportional Gain (K_p)**: 0.1
- **Derivative Gain (K_ds)**: 0.01

The control law is implemented as [9]:

$$u(t) = K_p \cdot e(t) + K_d \cdot \frac{de(t)}{dt} \quad (4)$$

where $e(t)$ represents the error signal at time t, and $u(t)$ is the steering command output.

The derivative term is computed using backward difference approximation:

$$\frac{de(t)}{dt} \approx \frac{e(t) - e(t-1)}{\Delta t} \quad (5)$$

where $\Delta t = 32$ ms represents the sampling period.

E. Multi-camera Management

The central camera maintains highest priority in the control hierarchy. When it detects at least one valid lane marking, it immediately assumes control authority. This design ensures that forward-looking visual data takes precedence whenever available, providing the most relevant guidance for immediate lane keeping.

When the central camera fails to detect lane markings (e.g., during sharp turns or temporary obstructions), the system evaluates peripheral cameras. The arbitration logic selects the side camera with the greater absolute error value, indicating more significant lane deviation detection. This approach prioritizes correction of larger errors while maintaining focus on the most critical navigational cues.

To prevent rapid switching and control conflicts between side cameras, the system implements a control locking mechanism. Once a side camera assumes control, it maintains authority until the central camera resumes lane detection. This stability measure ensures consistent steering behavior and prevents oscillatory control patterns that could compromise vehicle stability.

F. Trajectory Postprocessing

Following the real-time PD control phase, the collected GPS trajectory data undergoes postprocessing to analyze and optimize the vehicle's path. This offline analysis utilizes filtering techniques with the following parameters:

- **Savitzky-Golay Filter**: window_length = 13, polyorder = 3
- **Moving Average Window**: win_size = 3

- **Error Threshold**: 0.01

The postprocessing stage serves multiple purposes:

1) **Trajectory Smoothing**: Application of Savitzky-Golay [15] filtering with window length 13 and polynomial order 3 to smooth the collected GPS path data while preserving important trajectory features.
2) **Performance Analysis**: Evaluation of control system performance through analysis of the filtered trajectory data.
3) **Parameter Optimization**: Using the processed trajectory data to optimize control parameters for future iterations [10].
4) **Path Quality Assessment**: Threshold-based analysis (threshold = 0.01) to identify segments requiring control improvement.

This two-stage approach separates real-time control requirements from comprehensive performance analysis, allowing for robust online operation while enabling detailed offline optimization of the autonomous navigation system.

G. Velocity Profile

The objective of the profile is to provide a safe and dynamically feasible speed at each point of the path, taking into account lateral acceleration limits, vehicle longitudinal capabilities (acceleration/braking), and smoothness constraints [14].

1) Inputs and parameters:

- A sequence of trajectory nodes $p_i = (x_i, y_i)$, $i = 0 \dots N-1$, with inter-point distances $d_i = |p_{i+1} - p_i|$ (or a representative step ds when distances are uniform).
- Maximum allowed speed v_{max}.
- Maximum lateral acceleration a_{lat_max} (for example $a_{lat\_max} = \mu g$).
- Maximum longitudinal acceleration a_{accel} and braking deceleration magnitude a_{brake} (positive values).
- Safety factor $c_s \in (0, 1]$ to provide margin (typical values 0.8–0.95).

2) Curvature estimation: For a discrete trajectory the curvature κ_i at node i can be estimated using three consecutive points:

$$\kappa_i \approx \frac{2\,|(p_{i+1} - p_i) \times (p_i - p_{i-1})|}{|p_{i+1} - p_i|\,|p_i - p_{i-1}|\,|p_{i+1} - p_{i-1}|}. \tag{6}$$

The sign of κ_i indicates turn direction, while $|\kappa_i|$ is used to limit the speed.

3) Curvature-based speed limit: Limiting lateral acceleration yields a local maximum admissible speed:

$$v_{curv,i} = \sqrt{\frac{a_{lat\_max}}{|\kappa_i| + \varepsilon}}, \tag{7}$$

where ε is a small constant to avoid division by zero. The final local speed cap is

$$v_{limit,i} = c_s \cdot \min\{v_{max},\, v_{curv,i}\}. \tag{8}$$

4) Apex detection and turn segmentation: Turn segments are identified as continuous index intervals where $|\kappa_i|$ exceeds a threshold κ_{th}. The apex index of a turn is selected as

$$i_{apex} = \mathrm{argmax}_{i \in [istart, iend]}\, |\kappa_i|, \tag{9}$$

which typically corresponds to the point of maximum lateral demand and therefore the lowest local admissible speed inside the turn.

5) Velocity Profile Interpolation: The velocity points are interpolated along the full trajectory using Bezier Interpolation Method. This method creates smooth velocity transitions along a vehicle's path using mathematical curves [13]. The approach ensures gradual acceleration and deceleration rather than abrupt speed changes.

Input Parameters:

- **Path points**: $(x_{\text{turn}}, y_{\text{turn}})$ - coordinates defining the intended route;
- **Velocity limits**: v_{turn} - maximum safe speeds at each path point;
- **Current position**: $(x_{\text{start}}, y_{\text{start}})$ - vehicle's starting location;
- **Smoothing factor**: τ - controls how gradual the speed transitions are (0.1–0.3 for smooth, 0.7–0.9 for sharp);

Method Logic:

1) **Path distance calculation**: Compute cumulative travel distance s from start position through all path points;
2) **Intermediate points**: Automatically insert additional points between distant path locations to ensure smooth speed transitions;
3) **Bezier curve construction**: For each path segment between points, create a smooth velocity curve using four control points that define the speed profile;
4) **Velocity interpolation**: Calculate reference speed at any path position using the mathematical formula:

$$v(s) = (1-t)^3 v_0 + 3(1-t)^2 t v_1 + 3(1-t) t^2 v_2 + t^3 v_3$$

where t represents the normalized position along the current path segment.

Output Values:

- **Fine velocity profile**: v_{fine} - smooth speed values at 1000+ points along the entire path;
- **Path coordinates**: s_{fine} - corresponding distance measurements from start position;
- **Reference points**: s_{orig} - locations of original speed limits, s_{new} - locations of automatically added transition points;

The method ensures physically realistic acceleration patterns by enforcing smooth speed transitions, preventing sudden jerks that could destabilize vehicle control.

H. Path Following

The vehicle follows a precomputed path given as waypoints $p_i = (x_i, y_i)$ with reference speeds v_i. At each control step the nearest waypoint index i^* is found, and two errors are computed: cross-track error e_{ct} (signed lateral distance to

the path), speed error $e_v = v_{\text{ref}}(i^*) - v$. PID controller uses e_v to generate both lateral and longitudinal command (throttle/brake). This loop is executed every control cycle, which makes the vehicle track the reference path and velocity profile.

The following parameters were used for PID-controllers:

Cross-track error control:

1) $K_p = 0.2$
2) $K_d = 0.1$
3) $K_i = 0.01$

Velocity error control:

1) $K_p = 0.1$
2) $K_d = 0.01$
3) $K_i = 2$

IV. Experimental Results

The results of passing a race track are captured featuring the passed trajectory, velocity profile, and errors. We tested the Tesla Model 3 model performance on the Yas-Marina designed map using the Webots Simulator [12].

A. Camera Controller Map Exploration

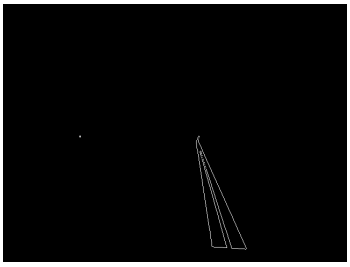

Fig. 1. Road markup detected by the edge detector

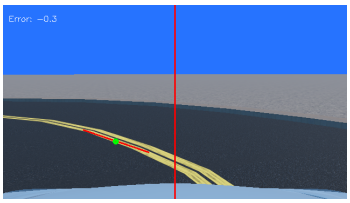

Fig. 2. Line markup's rotation detection

Performance testing in Webots simulation [12] demonstrates reliable lane following through curves and straight segments (Fig. 1, Fig. 2) at the target speed of 18 km/h, with the control arbitration logic successfully managing transitions between camera authorities. The PD controller parameters ($K_p = 0.1, K_d = 0.01$) provide an optimal balance between responsiveness and stability [8].

B. Adaptive Controller

As the trajectory points were captured, the post-processing pipeline produced smooth interpolation and the turn points were detected (Fig. 3).

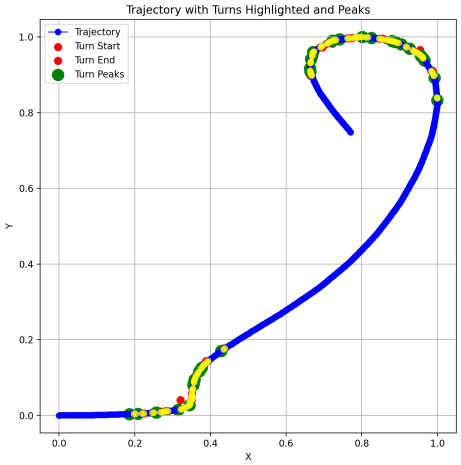

Fig. 3. Trajectory Post-Processing for Turn Detection on a Part of a Track

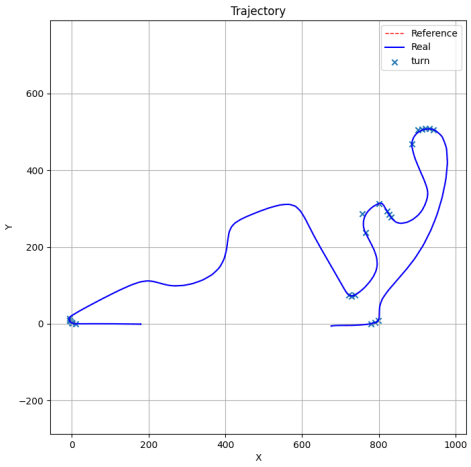

Fig. 4. Yas-Marina Trajectory with Detected Turns

Information about the location of turns was used to compute descriptive information for the model: the locations of the entry, apex, and exit for each turn, as well as associated velocities. In Fig. 4, the trajectory with placed turn points is demonstrated.

Based on the turn information, the velocity profile is interpolated to the rest of trajectory points. The vehicle successfully traverses along the trajectory with the intended velocity profile (Fig. 5).

The trajectory following is preserved fully with negligible errors of around 1 m, the velocity profile sometimes experiences minor overshoots of around 3 m/s but the overall quality is acceptable (Fig. 6). The system experiences a large overshoot at the end of the interpolation stage, which is natural, as the velocity is no longer controlled and the value is kept constant.

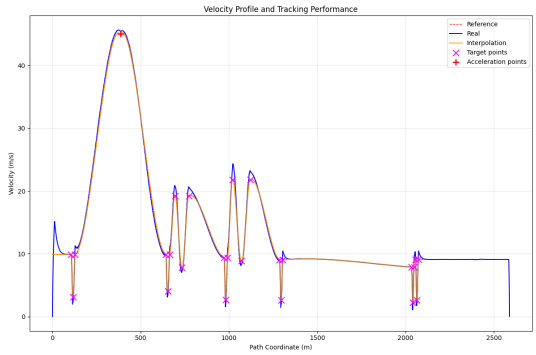

Fig. 5. Yas-Marina Track Velocity Profile

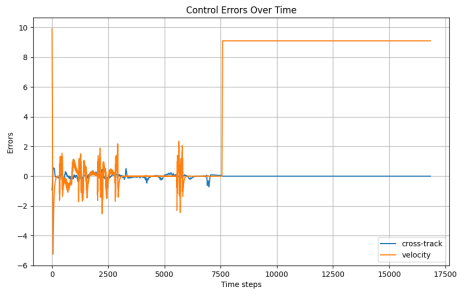

Fig. 6. Yas-Marina Track Cross-track and Velocity Errors

V. Analysis and Observations

Overall, the two-stage process achieves high performance results on the tested map. Moreover, the adaptive controller is also partially generalizable for any initial location with a constraint on recomputation of the velocity profile interpolation. Finally, the method is designed for the specific lane detection, and the approach can be employed on various maps having this feature.

VI. Conclusion

The priority-based multi-camera control system presented in this paper provides a robust solution for autonomous lane following. By combining hierarchical control authority with conflict prevention mechanisms and computational optimization through ROI processing, the system ensures reliable performance while leveraging the complementary strengths of multiple visual perspectives. The integrated PD controller [6], [7] generates precise steering commands that maintain vehicle stability while accurately tracking lane markings.

The deterministic arbitration strategy offers computational efficiency suitable for real-time applications while maintaining the safety benefits of sensor redundancy. The carefully tuned controller parameters ($K_p = 0.1, K_d = 0.01$) operating at 32 ms sampling rate provide optimal performance at the target vehicle speed of 18 km/h.

The adaptive controller preserves large velocity capabilities while keeping the car within the lane limits. The interpolation methods resulted in the reliable trajectory suitable for regulation via PID-contollers.

Future work will explore generalized lane detection or arbitrary navigation. Further exploration can be made in adaptive control gains for varying speed conditions [10] and integration with additional sensors for enhanced environmental perception, potentially incorporating advanced control strategies that combine predictive elements with proportional-derivative action [11].

References

[1] J. Canny, “A Computational Approach to Edge Detection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986.

[2] R. O. Duda and P. E. Hart, “Use of the Hough Transformation to Detect Lines and Curves in Pictures,” Communications of the ACM, vol. 15, no. 1, pp. 11-15, Jan. 1972.

[3] Y. Wang, E. K. Teoh, and D. Shen, “Lane detection and tracking using B-Snake,” Image and Vision Computing, vol. 22, no. 4, pp. 269-280, 2004.

[4] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, “Recent progress in road and lane detection: a survey,” Machine Vision and Applications, vol. 25, no. 3, pp. 727-745, 2014.

[5] K. J. Åström and T. Hägglund, “PID Controllers: Theory, Design, and Tuning,” Instrument Society of America, 1995.

[6] Åström et al., “Automatic tuning and adaptation for PID controllers - a survey,” Control Engineering Practice, vol. 1, no. 4, pp. 699-714, 1993.

[7] M. S. Ahmed, “A review of PID control, tuning methods and applications,” International Journal of Dynamics and Control, vol. 9, pp. 818-827, 2021.

[8] K. J. Åström and T. Hägglund, “Advanced PID Control,” ISA - The Instrumentation, Systems, and Automation Society, 2006.

[9] L. Da Costa et al., “PID Control as a Process of Active Inference with Linear Generative Models,” Entropy, vol. 21, no. 11, p. 1063, 2019.

[10] A. G. Ivanov, “Design and Implementation of Model Predictive Control Based PID Controller for Industrial Applications,” Energies, vol. 13, no. 17, p. 4422, 2020.

[11] J. J. Yamaura, “New approach of series-PID controller design based on modern control theory,” Results in Control and Optimization, vol. 14, p. 100349, 2025.

[12] Cyberbotics, ”Webots Open Source Robot Simulator,” cyberbotics.com. Accessed: Nov. 24, 2025. [Online.] Available: https://cyberbotics.com/

[13] R. T. Farouki, ”The Bernstein polynomial basis: A centennial retrospective, Computer Aided Geometric Design,” vol. 29, no. 6, pp. 379-419, 2012.

[14] R. Rajamani, ”Chapters 2, 3, 7: Lateral acceleration limits, curvature constraints, path tracking”, in *Vehicle Dynamics and Control,* Springer, 2011.

[15] A. Savitzky and M. J. E. Golay, ”Smoothing and Differentiation of Data by Simplified Least Squares Procedures,” *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, July 1, 1964.