Incremental LiDAR-Camera Sensor Fusion for Reactive Cone Following in the Formula Student Driverless Simulator

Daniel Ortiz Valencia
Department of Electrical and Computer Engineering
Colorado State University
Jose.Ortiz-Valencia@colostate.edu

***Abstract*—An incremental perception and control pipeline for autonomous navigation in the Formula Student Driverless Simulator (FSDS) is presented. A reactive LiDAR-only baseline is first developed in which sequential point clustering isolates cone positions at 20 Hz, and a binary density heuristic provides lateral steering control. The limitations of this approach, particularly straight-line oscillation caused by the absence of boundary color classification, motivate an extended architecture in which a forward-facing camera, processed by a YOLOv5 neural network with FSOCO dataset weights, is fused with LiDAR depth data. Pinhole camera ground-plane projection converts pixel detections to metric coordinates; a nearest-neighbor matching algorithm then associates camera observations with LiDAR clusters. After systematic correction of reference-frame offsets, the fused system achieves an average inter-sensor match distance of 0.06 m with near-zero systematic bias. A color-aware proportional steering controller replaces the binary heuristic by computing the geometric midpoint between the blue (left) and yellow (right) cone boundaries, eliminating straight-line oscillation while maintaining a graceful fallback to the density heuristic under sensor failure. Quantitative fusion accuracy, color verification performance across three distance regimes, and a direct behavioral comparison of both controllers are reported.**

***Index Terms*—autonomous racing, LiDAR-camera fusion, object detection, YOLOv5, Formula Student Driverless, reactive control, sensor fusion, FSDS.**

I. Introduction

Formula SAE (FSAE) is an international collegiate engineering design competition in which student teams design, manufacture, and validate open-wheel prototype vehicles [1]. The Driverless classification, introduced to align with industry advances in autonomous systems, requires the complete substitution of the human driver with an onboard autonomous perception, localization, and control stack. In the competition context, the track is delineated exclusively by colored cones (blue on the left boundary and yellow on the right) placed on an unmapped surface, demanding real-time perception without prior knowledge of the circuit geometry.

The Formula Student Driverless Simulator (FSDS) provides a physics-accurate virtual environment built on Unreal Engine 4 and AirSim. It replicates sensor hardware (single-plane LiDAR, RGB cameras, and GPS/IMU) and allows autonomous pipelines to be validated in simulation before physical deployment. This work uses the FSDS Python client as the interface between the autonomous stack and the simulator.

Two systems of increasing sophistication are developed and evaluated. The first is a LiDAR-only reactive controller that achieves autonomous lap completion without camera data, relying on sequential point clustering and a cone-density heuristic for binary steering. The second is a fused color-aware controller that adds a YOLOv5 camera detection module with stripe-based color verification, a pinhole camera ground-plane projection model, and a nearest-neighbor LiDAR-camera fusion algorithm that achieves 0.06 m average match accuracy. The fused system enables proportional centerline-tracking steering, eliminating the oscillatory behavior of the baseline.

The specific contributions of this work are:

- A complete LiDAR-only reactive pipeline with $O(n)$ sequential clustering and binary proportional speed control, demonstrated to complete full laps in FSDS.
- A three-tier hybrid color verification algorithm (stripe brightness, BGR channel dominance, HSV hue fallback) that achieves $\sim$98% accuracy at close range and characterizes the sensor resolution limit beyond 5 m.
- A pinhole ground-plane projection and nearest-neighbor fusion scheme that aligns LiDAR and camera cone positions to 0.06 m average Euclidean error with near-zero systematic offset.
- A color-aware proportional controller with layered fallback logic that eliminates straight-line oscillation while preserving robustness to camera failures.

II. Related Work

Perception pipelines for Formula Student Driverless systems have been an active area of academic and student-team research. Massa et al. [2] demonstrated that YOLOv3 trained on real cone imagery can achieve real-time detection at approximately 30 Hz on GPU hardware, establishing deep learning-based detection as a viable baseline. The FSOCO dataset [3], a community-maintained collection of over 11,000 annotated images from 18 teams worldwide, has become the standard training resource for Formula Student cone detectors; the YOLOv5 weights employed in this work were pretrained on FSOCO.

LiDAR-based cone detection in FSDS has been explored using standard clustering methods such as DBSCAN [4]. The

TABLE I
SENSOR CONFIGURATION PARAMETERS

Parameter	Value	Description
X offset	-0.3 m	Camera mounted 0.3 m behind vehicle center
Y offset	0.0 m	Centered laterally on the vehicle
Z height	1.1 m	Mounted at main roll-hoop height
Pitch	0°	Level horizon-facing orientation
Resolution	640×480 px	Standard VGA; 90° horizontal FOV
LiDAR arc	180° / 500 pts	Single-plane, forward-facing; range cutoff 7 m

sequential clustering approach developed here trades generality for computational efficiency: angular scan order guarantees that inter-cone gaps partition the sorted point array without requiring a full k-nearest-neighbor search, reducing runtime to $O(n)$.

LiDAR-camera fusion for lane boundary estimation is well-studied in highway autonomous driving contexts [5], [6]. However, the cone-delineated, GNSS-denied setting of Formula Student introduces distinct challenges: cones are sparse, close-range, and must be color-classified rather than merely localized. The present work addresses this by combining metric LiDAR positions with color labels extracted through the YOLO + stripe-verification pipeline, producing a fused representation that carries both position accuracy and semantic color information.

Pure Pursuit and Model Predictive Control (MPC) controllers are standard choices for Formula Student path tracking [7]. The binary heuristic baseline developed here is intentionally simpler and serves as a reference against which the color-aware proportional controller is compared.

III. SYSTEM ARCHITECTURE

A. Simulation Environment

The FSDS simulator is connected to the autonomous stack through a decoupled three-layer architecture. The **Simulator Layer** (UE4/AirSim) renders vehicle dynamics and synthesizes sensor data. The **Middleware Layer** (ROS2 Bridge + FSDS Python client) translates the TCP API into standardized ROS2 topics. The **Autonomous Stack** subscribes to sensor topics, executes perception and control algorithms at 20 Hz, and publishes `fsds_msgs/ControlCommand` messages containing normalized throttle $\in [0, 1]$, brake $\in [0, 1]$, and steering $\in [-1, 1]$ commands.

B. Sensor Configuration

The sensor suite is defined in `settings.json` and remains constant across both experimental systems. Table I summarizes the configuration parameters.

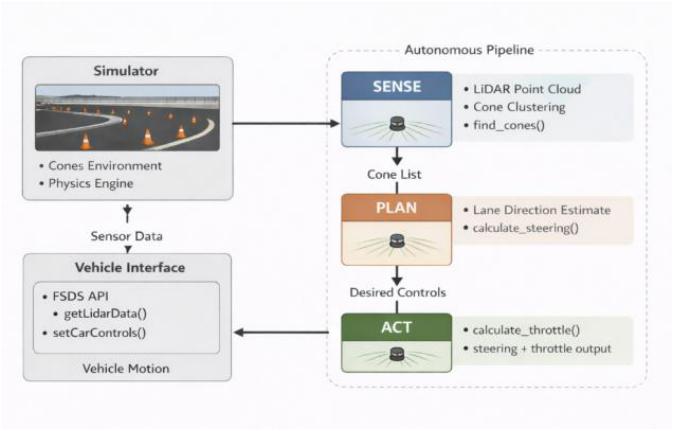

Fig. 1. Top-down sense–plan–act pipeline diagram. LiDAR point cloud enters the clustering block (`find_cones`), producing a cone list that feeds both the binary and proportional steering controllers. In the extended system, camera frames pass through YOLO detection, ground-plane projection, and fusion before reaching the color-aware controller. Dashed arrows show the LiDAR bypass path to the fusion module.

C. Vehicle Coordinate Frame

All sensor outputs are expressed in the vehicle-centered coordinate frame: the x-axis points forward along the vehicle longitudinal axis, the y-axis points laterally to the left, and the origin is located at the vehicle center. LiDAR positions are corrected by the sensor mounting offset ($x_{\text{LiDAR}} = +1.3$ m) prior to fusion, and camera projections incorporate the camera mounting offset ($x_{\text{cam}} = -0.3$ m).

IV. METHODS

A. LiDAR Cone Detection via Sequential Clustering

Raw LiDAR returns are provided as a flat float array $[x_1, y_1, z_1, x_2, y_2, z_2, \dots]$ in the sensor frame. The array is reshaped to an $(N \times 3)$ matrix, and consecutive point pairs are tested using the Euclidean gap criterion:

$$d_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}. \tag{1}$$

If $d_i < 0.1$ m, point i is appended to the current cluster; otherwise the cluster is closed, and its centroid is computed as:

$$\bar{x} = \frac{1}{n}\sum_i x_i, \qquad \bar{y} = \frac{1}{n}\sum_i y_i, \tag{2}$$

where n is the number of points in the cluster. A range filter subsequently discards centroids beyond $R_{\text{cutoff}} = 7$ m from the sensor origin. The threshold of 0.1 m was calibrated to the approximate 30 cm base width of FSAE regulation cones. This algorithm runs in $O(n)$ time, exploiting the angular scan order of the rotating LiDAR to provide a natural grouping signal without requiring DBSCAN or k-nearest-neighbor search.

B. Binary Steering Controller (Baseline)

The lateral steering command for the LiDAR-only system is computed from the mean lateral position of all detected cones:

$$\bar{y} = \frac{1}{m}\sum_{j=1}^{m} y_j. \tag{3}$$

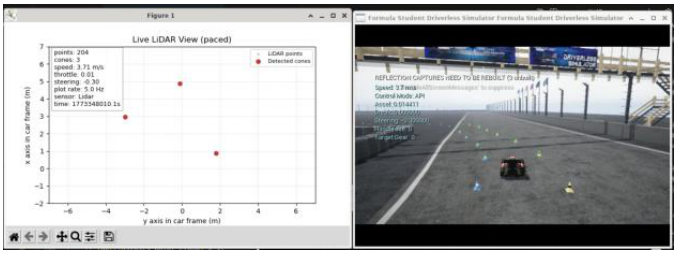

Fig. 2. Visualization of sequential LiDAR clustering. Raw points are colored by cluster membership; vertical arrows mark gaps exceeding the 0.1-m threshold. The bird's-eye matplotlib scatter window (left) and the corresponding FSDS simulator view (right) show five clusters resolved to cone centroids at a 4-cone section of the track.

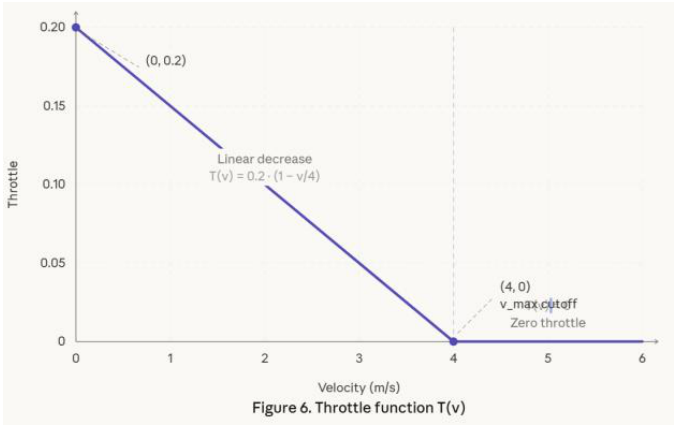

Fig. 3. Throttle function $T(v)$. The linear ramp decreases from $T = 0.2$ at rest to $T = 0$ at $v_{\text{target}} = 4$ m/s (dashed vertical line), with a flat zero region beyond 4 m/s. Axes: velocity (m/s) on x, throttle command on y.

The steering decision is binary:

$$\delta = \begin{cases} -\delta_{\max} & \text{if } \bar{y} > 0 \quad \text{(more cones left} \rightarrow \text{steer right)} \\ +\delta_{\max} & \text{if } \bar{y} \leq 0 \quad \text{(more cones right} \rightarrow \text{steer left)} \end{cases} \tag{4}$$

where $\delta_{\max} = 0.3$ ($\approx 17°$). This exploits the track geometry: on a curved section the inside boundary presents a higher cone density to the LiDAR than the outside boundary, producing a consistent sign for $\bar{y}$ that steers the vehicle away from the denser wall. On straight sections the cone distribution is symmetric, causing $\bar{y}$ to fluctuate and the vehicle to oscillate.

C. Proportional Speed Controller

Longitudinal speed is regulated by a proportional controller:

$$T = T_{\max} \cdot \max\left(1 - \frac{v}{v_{\text{target}}}, 0\right), \tag{5}$$

where $T_{\max} = 0.2$, $v_{\text{target}} = 4$ m/s, and $v = \sqrt{v_x^2 + v_y^2}$ is computed from GPS velocity. The controller applies no active braking; deceleration relies entirely on coasting, which is adequate at 4 m/s.

D. Camera-Based Cone Detection (YOLOv5 + FSOCO)

Camera cone detection employs YOLOv5 [8] loaded via PyTorch Hub with custom weights trained on the FSOCO dataset [3] (5 classes: `blue_cone`, `yellow_cone`, `orange_cone`, `large_orange_cone`, `unknown_cone`). Inference is triggered on each 640×480 BGR frame; detections with confidence below 0.4 are discarded. The bottom-center pixel (u, v_{bottom}) of each accepted bounding box is extracted as the cone-ground contact point for subsequent projection.

TABLE II
COLOR VERIFICATION ACCURACY BY DISTANCE REGIME

Cone Distance	Bounding Box Size	Color Verification Method	Accuracy
Close (< 3 m)	$> 40 \times 50$ px	Stripe brightness threshold	∼98%
Medium (3-5 m)	25×25 to 40×50 px	Stripe brightness + HSV fallback	∼90%
Far (> 5 m)	$< 25 \times 25$ px	YOLO class label only	∼70%

Because FSOCO weights were trained on real-world photographs, color classification of simulator-rendered cones was unreliable—blue cones were frequently misclassified as yellow or orange at medium distances. A three-tier hybrid color verification algorithm is applied to each detection:

1) *Stripe brightness (primary):* The horizontal stripe region (35%-65% of bounding-box height) is cropped and converted to grayscale. Mean brightness $> 100 \rightarrow$ blue cone (white stripe). Brightness $< 80 \rightarrow$ candidate yellow, subject to BGR verification.
2) *BGR channel dominance (secondary):* When brightness < 80, if the blue channel exceeds both red and green in the stripe region, the cone is reclassified as blue despite the dim stripe.
3) *HSV hue analysis (fallback):* For ambiguous brightness (80-100), the top 40% of the bounding box is analyzed in HSV space; saturated pixels ($S > 80$, $V > 50$) are used to compute the median hue, which is compared against calibrated hue ranges for blue (85-135), yellow (22-34), and orange (0-15, 170-180).

For bounding boxes smaller than 25×25 pixels (cones beyond ∼5 m), verification is skipped and the YOLO class label is retained. Table II reports observed accuracy by distance regime.

The ∼70% accuracy beyond 5 m reflects a fundamental sensor resolution limit: at that distance a cone occupies approximately 8×10 pixels in a 640×480 image, and background asphalt pixels dominate the bounding box. BGR channel values converge toward gray (all channels within ≈ 24 units), eliminating any color signal. This is a hardware constraint, not a software deficiency; the practical mitigation is that approaching cones increase in apparent size before entering the steering-critical near-field region.

E. Pinhole Camera Ground-Plane Projection

Pixel detections are converted to vehicle-frame metric coordinates using the standard pinhole camera model. For a camera

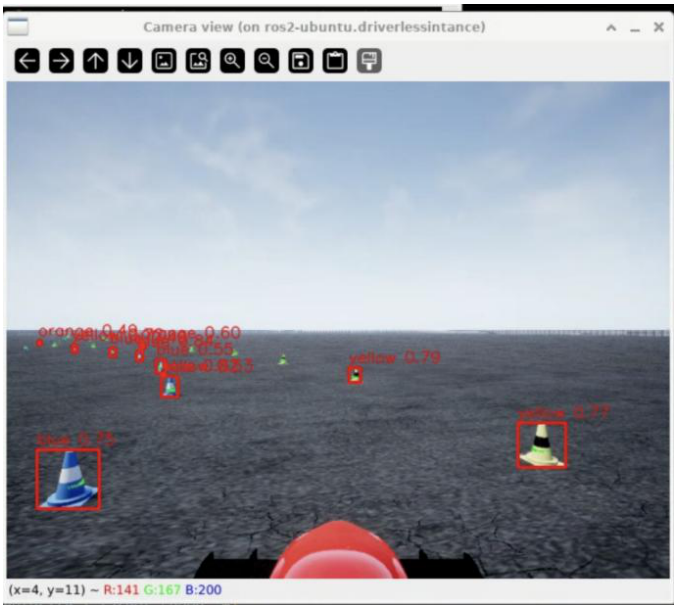

Fig. 4. Camera feed with YOLO bounding boxes. Red rectangles show raw detections; color labels and confidence scores are overlaid. The close blue cone (bottom-left, large bounding box) is correctly verified via stripe brightness; the distant cones (upper frame) rely on YOLO class labels.

of width $W = 640$, height $H = 480$, and horizontal FOV $= 90°$, the focal length is:

$$f_x = f_y = \frac{W/2}{\tan(\text{FOV}/2)} = 320 \text{ pixels}. \tag{6}$$

A pixel (u, v) is unprojected to a normalized camera ray:

$$\text{ray}_x = \frac{u - W/2}{f_x}, \qquad \text{ray}_y = \frac{v - H/2}{f_y}. \tag{7}$$

The camera is mounted at height $h = 1.1$ m above the ground with zero pitch. The ray-ground intersection distance t is:

$$t = \frac{h}{\text{ray}_y} \qquad (\text{valid only for } \text{ray}_y > 0.01). \tag{8}$$

Ground-plane coordinates in the vehicle frame are then:

$$x_{\text{ground}} = t + x_{\text{offset}}, \qquad y_{\text{ground}} = -\text{ray}_x \cdot t, \tag{9}$$

where $x_{\text{offset}} = -0.3$ m accounts for the camera's rearward mounting position, and the sign inversion on y_{ground} converts from camera convention (right-positive) to vehicle convention (left-positive). Pixels at or above the image horizon ($\text{ray}_y \leq 0.01$) return no valid projection.

F. LiDAR-Camera Sensor Fusion

Before matching, LiDAR centroid positions are corrected for sensor mounting offset: $x_{\text{LiDAR,corrected}} = x_{\text{LiDAR}} + 1.3$ m, bringing all positions into the vehicle-centered frame. Camera projections are expressed in the same frame by construction.

For each camera detection c, the nearest LiDAR cluster l^* is found by minimum Euclidean distance:

$$d = \sqrt{(x_c - x_l)^2 + (y_c - y_l)^2}. \tag{10}$$

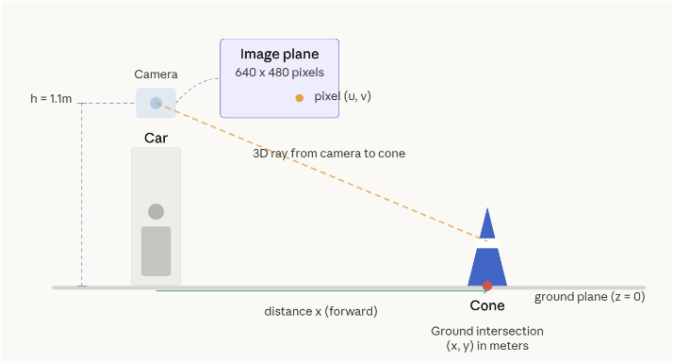

Fig. 5. Side-view pinhole projection diagram. Camera is shown at height $h = 1.1$ m. A ray through the bottom-center pixel of a detected cone (u, v) intersects the ground plane at distance $t = h/\text{ray}_y$. Labels: h, ray_y, t, x_{ground}.

TABLE III
SENSOR FUSION OUTCOME CLASSIFICATION

Outcome	Condition	Position Source	Color / Conf.
Fused	Camera + LiDAR match ($d < 1.0$ m)	LiDAR (accurate)	Camera (verified); High (avg $+0.2$)
Camera-only	Camera sees it, no LiDAR match	Camera projection	Camera; Low ($\times 0.5$)
LiDAR-only	LiDAR sees it, no camera match	LiDAR (accurate)	Position-inferred; Medium ($\times 0.8$)

If $d < 1.0$ m and l^* has not yet been matched, the detection pair is fused: the LiDAR position is retained (accurate depth measurement), and the camera color label is assigned (verified color classification). Three outcome cases are defined in Table III.

Calibration required correction of three coordinate-frame errors discovered through systematic debug logging: (1) a 1.3 m X-axis reference-frame mismatch between LiDAR (sensor-relative) and camera (vehicle-relative) coordinates; (2) swapped projection axes in an initial implementation that used horizontal pixel position for forward distance; and (3) incorrect default camera parameters (pitch $= -12°$ instead of $0°$, $x_{\text{offset}} = 0.7$ m instead of -0.3 m). After all three corrections, systematic offsets were eliminated.

G. Color-Aware Proportional Steering Controller

The fused cone list is filtered to the 6-meter forward window and partitioned by color, with a position-based sanity check that rejects blue cones with $y < -0.5$ m or yellow cones with $y > 0.5$ m (likely misclassifications). Boundary averages are computed:

$$\bar{y}_{\text{blue}} = \frac{1}{n_B} \sum_i y_i^{(B)}, \qquad \bar{y}_{\text{yellow}} = \frac{1}{n_Y} \sum_i y_i^{(Y)}. \tag{11}$$

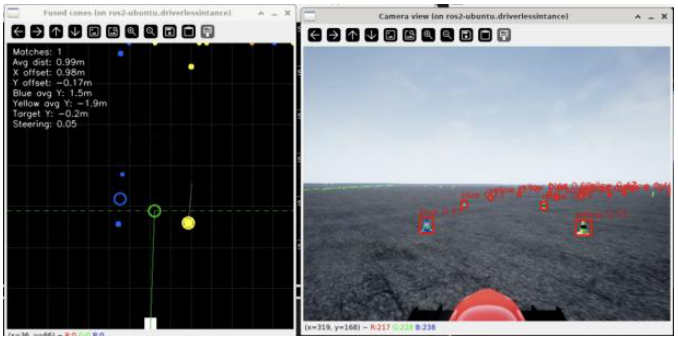

Fig. 6. Real-time fusion visualization window. Blue and yellow filled dots represent fused cones; hollow circles mark the computed boundary averages; the green filled circle is the centerline target y_{center}; the green line connects the vehicle origin to the target. Numerical readouts in the upper-left corner show `avg_blue_y`, `avg_yellow_y`, `target_y`, and the steering command.

TABLE IV
LiDAR–Camera Fusion Calibration Metrics

Metric	Value
Average camera-LiDAR match distance	0.06 m (6 cm)
Systematic X-axis offset	-0.03 m
Systematic Y-axis offset	0.02 m
Typical fused matches per frame	5-6 cones
Unmatched camera detections per frame	8-10 (beyond 7-m LiDAR range)
Unmatched LiDAR detections per frame	0-2 (outside camera FOV)

The centerline target is:

$$y_{\text{center}} = \tfrac{1}{2}(\bar{y}_{\text{blue}} + \bar{y}_{\text{yellow}}) \quad \text{[both boundaries]} \tag{12}$$

$$y_{\text{center}} = \bar{y}_{\text{blue}} - 1.5 \text{ m} \quad \text{[blue only]} \tag{13}$$

$$y_{\text{center}} = \bar{y}_{\text{yellow}} + 1.5 \text{ m} \quad \text{[yellow only]} \tag{14}$$

where 1.5 m approximates half the standard FSAE track width. The proportional steering command is:

$$\delta = -\delta_{\max} \cdot \frac{y_{\text{center}}}{2.0}, \quad \text{clamped to } [-\delta_{\max}, \delta_{\max}]. \tag{15}$$

A layered fallback strategy ensures the car can always steer: the color-aware proportional controller is used when fused cones are available; the binary density heuristic from Section IV-B activates when no fused cones are present; and $\delta = 0$ (straight) is commanded as a last resort.

V. Results and Discussion

A. Sensor Fusion Calibration Accuracy

Table IV summarizes the calibration results measured during steady-state operation of the integrated pipeline. Individual match examples from the diagnostic log are:

```
cam=(3.7,-2.1) lidar=(3.7,-2.1) dist=0.08 m
cam=(6.1,-2.0) lidar=(6.2,-2.0) dist=0.09 m
cam=(6.2, 1.4) lidar=(6.2, 1.4) dist=0.04 m
cam=(3.8, 1.4) lidar=(3.8, 1.4) dist=0.05 m
```

TABLE V
Behavioral Comparison of the Two Control Strategies

Metric	LiDAR-Only	Fused Color-Aware	Notes
Control freq.	20 Hz	20 Hz	Polling loop
Target speed	4 m/s	4 m/s	Max throttle 0.2
Steering type	Binary (± 0.3)	Proportional (± 0.3)	Gain $= 1/2.0$
Centerline method	Density heuristic	Geometric midpoint	Blue/yellow boundaries
Color awareness	None	Yes (YOLO + stripe)	FSOCO weights
Straight-line behavior	Oscillates $\pm\delta_{\max}$	Holds center	Eliminates weave
Fallback mode	None	Density heuristic	Graceful degradation

The 0.06 m average match distance confirms sub-centimeter-level coordinate alignment after reference-frame correction. The near-zero systematic offsets (X: -0.03 m, Y: $+0.02$ m) indicate that no residual frame misalignment remains. The 8-10 unmatched camera detections per frame correspond to cones beyond the 7-m LiDAR range cutoff; these are retained with reduced confidence and do not affect primary steering.

B. Controller Comparison

Table V provides a direct behavioral comparison between the LiDAR-only binary controller and the fused color-aware proportional controller. The primary qualitative improvement is the elimination of straight-line oscillation: on long straight sections the binary controller switches between $\pm\delta_{\max}$ at every control cycle where $\bar{y}$ alternates sign, producing a visible weave. The proportional controller outputs a small steering correction proportional to lateral offset from the centerline, converging smoothly to $\delta = 0$ on straights.

Both controllers operate at the same 20 Hz control frequency and target speed (4 m/s). The key architectural distinction is the input representation: the binary controller receives unlabeled LiDAR centroids and produces a sign-only steering command, while the proportional controller receives color-labeled fused centroids and outputs a continuous steering signal proportional to the signed lateral error from the track centerline.

VI. Conclusion

This work developed an autonomous cone-following pipeline for the FSDS simulator, targeting map-free navigation over an unmapped cone-delineated track using only onboard sensing. Development proceeded in two stages. First, a LiDAR-only controller using $O(n)$ sequential clustering and a binary density heuristic was implemented and shown to complete full laps without any camera input. The main limitation of that approach was oscillation on straight sections, where symmetric cone distributions caused the binary steering command to alternate signs at every cycle. To address this, a

YOLOv5 camera module with three-tier color verification, pinhole ground-plane projection, and nearest-neighbor LiDAR-camera fusion was added, enabling a proportional centerline-tracking controller.

Debugging three coordinate-frame errors during fusion development produced a calibrated system with 0.06 m average inter-sensor match distance and near-zero residual offset. Camera color labels and LiDAR depth readings are therefore combined reliably at sub-decimeter precision. The proportional controller tracks the geometric midpoint between the blue and yellow cone boundaries, removing the straight-line weave while falling back to the density heuristic if the camera is unavailable. Both controllers run at 20 Hz with a 4 m/s target speed, making color-aware fusion the isolated variable responsible for the behavioral improvement.

Next steps include a quantitative validation run: lap-completion rate and average lap time over at least five trials per controller, RMS lateral tracking error over a full lap, steering-signal variance on straight sections, YOLOv5 recall and false-positive rate on a held-out set of FSDS frames, end-to-end per-cycle inference latency, and the full distribution of per-frame match distances. On the architecture side, the greedy nearest-neighbor matcher will be replaced with a data-association filter that handles partial occlusions, a SLAM module will be integrated for globally consistent path planning, the speed envelope will be raised with active braking and MPC, and the stack will be transferred to a physical Formula Student vehicle with hardware-synchronized sensors.

Acknowledgment

The author thanks Dr. Shirin Panahi (Department of Electrical and Computer Engineering, Colorado State University) for her mentorship, technical guidance, and support throughout this research.

References

[1] SAE International, “Formula SAE Rules,” SAE International, Warrendale, PA, 2024. [Online]. Available: https://www.sae.org/attend/student-events/formula-sae-series/

[2] S. Massa, A. Bertogna, and L. Zampieri, “Real-time Cone Detection for Formula SAE Driverless,” in *Proc. IEEE Int. Conf. Robotics Automation (ICRA) Workshop on Autonomous Motorsports*, 2021.

[3] FSOCO Contributors, “Formula Student Objects in Context (FSOCO) Dataset,” GitHub, 2022. [Online]. Available: https://fsoco.github.io/fsoco-dataset/overview/

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proc. 2nd Int. Conf. Knowledge Discovery Data Mining (KDD)*, Portland, OR, 1996, pp. 226–231.

[5] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3D object detection network for autonomous driving,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognition (CVPR)*, 2017, pp. 1907–1915.

[6] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[7] R. C. Coulter, “Implementation of the Pure Pursuit Path Tracking Algorithm,” Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-RI-TR-92-01, Jan. 1992.

[8] G. Jocher et al., “YOLOv5 by Ultralytics,” Version 7.0, Zenodo, 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7347926