A Visual Mapping Pipeline for Traffic Cone Detection and Persistent Global Mapping in Autonomous Racing

Filippo Gibertini, Matteo Gombia, Leonardo Nels
(300226@studenti.unimore.it) (302715@studenti.unimore.it) (270099@studenti.unimore.it)

University of Modena and Reggio Emilia, Modena, Italy

We present a visual mapping pipeline tailored for autonomous systems participating in autonomous racing competitions. The system detects traffic cones and registers their locations in a persistent global map that an autonomous vehicle must navigate. The proposed pipeline comprises: (i) a yolov11-m network fine-tuned for robust cone detection under dynamic track conditions; (ii) a visual inertial SLAM module that estimates the vehicle's pose and twist in real time; (iii) a geometric depth-map generator that leverages a stereo camera setup; (iv) a fusion layer that computes a 3D vector originating at the camera center, directed towards the center pixel of each detected cone, with a magnitude equal to the estimated depth, to accurately determine the cone's position in the real world; and (v) a EKF-based mapping component that combines vehicle odometry and cone positions to build and continually update the global map. This approach enables efficient and reliable localization and mapping in the dynamic environment of autonomous racing.

0 INTRODUCTION

Autonomous driving technologies have seen significant advances in recent years, driven by innovations in perception, localization, and planning. Among the prominent platforms fostering such developments there are academic competitions like the Formula Student Driverless (FSD) challenges [1], which challenges student teams with designing and deploying fully autonomous race cars capable of navigating dynamic tracks marked by traffic cones. These competitions simulate real-world constraints and demand robust, real-time solutions for perception and mapping in the face of high-speed operation, variable lighting, and tight resource budgets.

A critical requirement in FSD events is the ability to reliably detect and localize traffic cones of different colors, which delineate the race track, and to maintain an accurate global map of their positions. This mapping capability is essential not only for track exploration during autonomous track drive (ATD) stages but also for enabling optimized trajectory planning during subsequent runs. However, achieving this in a racing context introduces unique challenges: high vehicle dynamics increase perception latency requirements; stereo and monocular vision sensors are affected by motion blur and varying lighting conditions; and odometry drift over longer laps can corrupt map consistency if not addressed.

To tackle these challenges, we propose a visual mapping pipeline specifically tailored for Formula Student.

In this paper, we describe the design and implementation of the proposed pipeline, evaluate its performance in simulated and real-world FSD environments, and discuss its strengths and limitations.

1 METODOLOGY

The system integrates: (i) a YOLOv11 [2] object detector, fine-tuned for fast and reliable cone detection under racing conditions; (ii) a visual inertial SLAM [3] module for real-time pose estimation; (iii) a geometric depth-map generator that leverages stereo imagery for depth recovery; (iv) a fusion layer that computes a 3D vector originating at the camera center, directed towards the center pixel of each detected cone, with a magnitude equal to the estimated depth, to accurately determine the cone's position in the real world; and (v) a EKF-based [4] mapping component that combines vehicle odometry and cone detections to construct and maintain a persistent global map. This modular architecture enables high accuracy in cone localization and robust map updates even in challenging racing scenarios. The whole pipeline is deployed leveraging ROS2 [5] middle-ware which provides a lightweight publisher/subscriber topology to allow all the components of the pipeline to communicate effectively and in real time.

1.1 Cones detection using YOLOv11

Reliable cones detection on the camera frames is a critical first step in the pipeline. We employ a YOLOv11-m architecture, selected for its balance between detection accuracy and real-time performance on embedded computing platforms. The network is finetuned on the Formula Student Objects in Context Dataset (FSOCO)[6] which provides a diverse collection of annotated images (approximately 11,000) from test tracks. The dataset provides five different classes: blue cone, yellow cone, small orange cone, large orange cone, other, and covers a wide range of environmental conditions, including variable illumination, shadows, partial occlusions, and motion blur, making it well-suited for the operational domain. To improve model robustness and generalization, we applied a suite of augmentation techniques during training. Specifically, we configured the augmentation pipeline with a scale factor of 0.2 and shear angle of 4 degrees to account for perspective changes from varying vehicle orientations. Photometric augmentations included hue, saturation, and value adjustments (hsv_h=0.05, hsv_s=0.05, hsv_v=0.15) to simulate lighting variations. Additionally, mixup (0.10) and copy-paste (0.10) were incorporated to synthetically enrich the dataset with composite scenes, enhancing the detector's resilience to overlapping cones and complex backgrounds. At inference time, the detector subscribes to rectified RGB stereo camera images as inputs and publishes bounding boxes, confidence scores, and class labels. Post-processing steps include non-maximum suppression to suppress duplicate detections and a confidence threshold to filter out low-probability outputs. The results were the following:

Metric	Value
Precision	0.844
Recall	0.795
mAP@50	0.801
mAP@50-95	0.573
Val Box Loss	0.725
Val Cls Loss	0.318

Table 1.

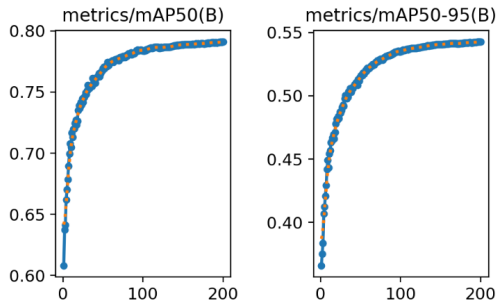

Fig. 1. Example of an image included in the AES paper.

1.2 Depth-map generation

Generating an accurate depth map for each pixel is a critical step in the pipeline. We leverage a stereo camera setup, which follows these steps: (i) acquire synchronized left and right images; (ii) perform image rectification to align the views so that corresponding points lie along the same horizontal scan-line; (iii) compute the disparity d as the horizontal difference in pixel coordinates between matched points:

$$d = u_L - u_R \tag{1}$$

where u_L and u_R are the horizontal image coordinates in the left and right images, respectively; (iv) compute the depth z of each pixel using the stereo triangulation formula:

$$z = \frac{f \cdot B}{d} \tag{2}$$

Here, f is the focal length of the camera (in pixels), and B is the baseline, the distance between the two camera centers. This process yields a dense depth map in which each pixel encodes the distance from the camera to the observed scene point.

1.3 Fusion module

After obtaining the bounding boxes (bb) of each detected cone and generating the depth map, it is possible to fuse that information in order to obtain the real-world positions of the cones in the reference frame of the car in a given instant. For each detected cone, the first step is to decide which pixel of bb to pick, the choices could be the middle point pixel of the base of the bb or the center pixel of the bb. In this implementation, the base pixel is taken because we are interested in the points that lie on the ground. This choice has also been validated with experimental tests of the precision of the output. The second step is to find the correspondent value of the pixel in the depth map which is trivial due to the fact that the image and the depth-map share the same dimension. The third step is to use the intrinsic parameters of the camera to reconstruct the 3D position of the pixel $[XYZ]$ in the camera frame:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad P_{\text{image}} = (u, v), \quad \text{Dist.} = z \tag{3}$$

We can set up a system to find the real-world coordinates of the cone in the camera frame:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = z K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{4}$$

The solutions are:

$$X = \frac{(u - c_x) \cdot z}{f_x}, \quad Y = \frac{(v - c_y) \cdot z}{f_y}, \quad Z = z \tag{5}$$

Now we account for the camera extrinsic to convert from camera frame to car frame:

$$T_{\text{cam}\to\text{car}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix} \tag{6}$$

Leveraging homogeneous coordinates, we finally obtain the point coordinates in the car frame:

$$P_{car}^{(4D)} = T_{\text{cam}\to\text{car}}^{4x4} \cdot P_{cam}^{(4D)} \tag{7}$$

These coordinates will later undergo further conversion from the car frame to the global frame.

1.4 Visual-Inertial SLAM

To estimate vehicle odometry (pose, twist) in real time, we integrate a Visual-inertial Simultaneous Localization and Mapping (VI-SLAM) module into the pipeline. This component fuses visual features from the stereo camera and inertial data from the integrated Inertial Measurement Unit (IMU) to provide robust odometry estimation even under challenging dynamics and partial occlusions. VI-SLAM is a widely used approach for estimating the six degrees-of-freedom (6-DoF) pose of a moving platform by combining visual information from cameras with inertial measurements from an IMU. In general, a VI-SLAM system operates in two stages: a front-end, which processes raw sensor data to detect and track visual features, and a back-end, which integrates these observations to estimate the trajectory of the platform and build a sparse map of the environment.

In the front-end, keypoints are extracted from incoming image frames and matched across consecutive frames to track motion. Stereo vision provides depth information by triangulating matched features from the left and right images. Meanwhile, IMU data are used to infer short-term motion between frames, offering robustness in scenarios with low-texture regions or motion blur where purely visual tracking may fail.

The back-end employs a filtering approach to fuse visual and inertial data, producing pose estimates that are smoother and more consistent than those obtained from visual or inertial odometry alone. In outdoor environments, such as automotive applications and, specifically, Formula Student competitions, there is also the possibility of integrating GNSS data into the fusion process. This can be particularly beneficial in scenarios like racetracks, where visual keypoints may be insufficient for robust odometry due to repetitive patterns or textureless areas. However, in our prototype setup, we did not have access to the RTK-GPS system available on our full-scale autonomous vehicle. Such an integration could further improve the overall accuracy of the vehicle's pose estimation.

This approach allows to keep a consistent trace of the odometry of the car in the global frame over time.

1.5 EKF Mapper

Once the cone positions have been computed in the car reference frame (Section 1.3) and the vehicle pose estimated via the visual inertial SLAM system (Section 1.4), these two data sources are fused within an Extended Kalman Filter (EKF) to maintain a consistent and persistent global map of the traffic cones.

The EKF is a recursive state estimator well suited to problems involving nonlinear motion models and noisy measurements, making it ideal for fusing high-frequency odometry with sparse but spatially informative cone observations. In our implementation, the EKF maintains a dynamic state vector consisting of all currently observed cones in the global frame, each represented by its 2D position (x,y) and an associated covariance array. The cones are assumed to remain static, and the filter focuses on refining their positions as new detections are made.

At each time step, the filter receives:

- A pose update from the VI-SLAM system, providing the current vehicle pose $T_{car\to world}$ in the global frame.
- A set of cone detections in the car frame.

These transformed cone positions serve as observations in the EKF update step. A nearest-neighbor data association strategy is used to match new detections to existing cones based on Euclidean distance and class consistency (e.g., blue vs. yellow cones). If a match is found, the EKF uses the observation to update the landmark's state and reduce its uncertainty. If no match is found, a new landmark is initialized in the filter state.

To reduce drift and suppress false positives, a confidence score is associated with each landmark. This score is incremented for each consistent detection and decremented when the landmark is not observed over multiple time steps. Landmarks with persistently low confidence are eventually pruned off the map.

This filtering framework allows the global cone map to evolve over time as the vehicle completes multiple laps, gradually refining cone positions and compensating for noise in detection and pose estimation.

1.6 Bird-Eye-View Generation

To improve interpretability and facilitate debugging of the perception pipeline, we implemented a bird's-eye-view (BEV) visualization module. This module transforms the perspective view from the forward-facing camera into a top-down representation of the scene, aligning the horizontal plane (typically the ground) with the image plane. The BEV output provides a clearer spatial understanding of cone positions relative to the vehicle, which is especially useful for assessing detection consistency and map quality.

The transformation relies on a homography array $\mathbf{H}$, which projects points from the 2D image plane onto the ground plane under the assumption that all cones lie on a flat surface at $Z = 0$. This assumption is valid for most Formula Student Driverless tracks, which are smooth and level.

The homography is computed using the known camera intrinsic parameters $\mathbf{K}$ and the extrinsic calibration parameters $(\mathbf{R},\mathbf{t})$, following the planar projection relation:

$$\mathbf{H} = \mathbf{K}\,[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] \tag{8}$$

Here, $\mathbf{r}_1$ and $\mathbf{r}_2$ are the first two columns of the rotation array $\mathbf{R}$, corresponding to the camera's orientation with respect to the world frame, and $\mathbf{t}$ is the translation vector. This formulation effectively constructs a 3×3 transforma-

tion array that relates the pixel coordinates to the positions on the ground.

Once computed, the homography $\mathbf{H}$ is applied to the image using opencv `warp2perspective` and `imwarp` functions from cv2 [7], yielding a warped image that represents the scene from a top-down perspective. The output resolution and scale are chosen to reflect real-world distances (e.g., meters per pixel), allowing consistent visualization across runs.

This BEV module plays a critical role in visual validation of the perception system. It enables inspection of cone positions in map space, supports landmark association checks, and provides immediate visual feedback on camera calibration accuracy and geometric transformations.

Fig. 2. Camera view

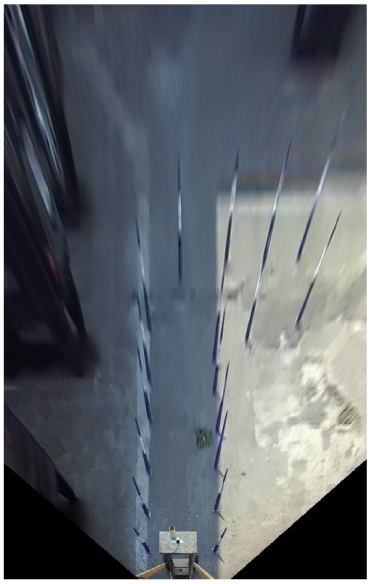

Fig. 3. Bird-Eye-View

2 EXPLORATORY WORK AND DISCARDED APPROACHES

During development of the visual mapping pipeline, several techniques and configurations were explored but ultimately not included in the final system due to either performance limitations, lack of robustness, or integration complexity. Documenting these attempts is important for understanding the design trade-offs that shaped the final implementation.

2.1 Monocular Image Approach

An alternative approach explored during the early stages of development was to estimate the positions of the cones using a single monocular image assuming that all the cones lie on a flat ground plane, specifically $Z = 0$ in the world coordinate system. Under this assumption, it is possible to project a ray from the camera optical center through each pixel coordinate (u, v) and compute the 3D location where this ray intersects the ground plane. This approach does not require stereo depth information.

Mathematically, this approach involves back-projecting the image point in a 3D direction using the inverse of the intrinsic array of the camera $\mathbf{K}^{-1}$, and then solving for the intersection point between the resulting ray and the plane $Z = 0$.

In practice, we found that this method was extremely sensitive to changes in vehicle pitch. During longitudinal acceleration and braking, the vehicle's chassis experiences pitch dynamics, tilting slightly forward or backward. These changes alter the effective orientation of the camera, violating the assumption of a fixed projection geometry and leading to systematic errors in cone position estimates.

Due to these limitations and the lack of robustness under real-world driving conditions, this monocular projection approach was discarded in favor of a stereo-based method, which directly computes depth for each pixel and is more resilient to camera motion and terrain variability.

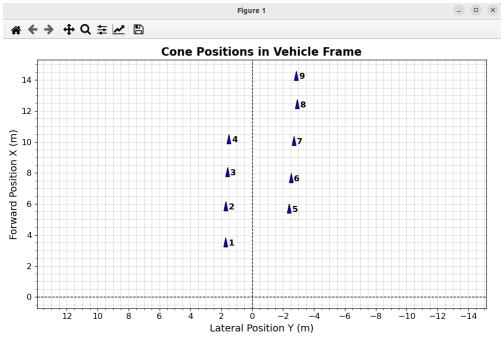

Fig. 4. Projection of cones on ground plane

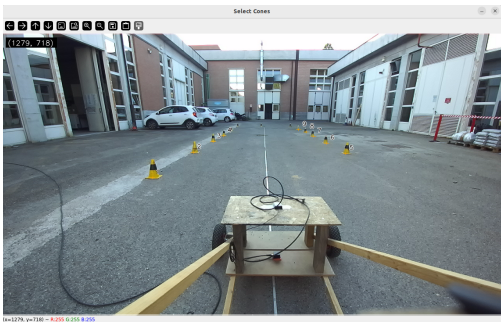

Fig. 5. Graphical interface

2.2 Stereo-images point-cloud generator

Another approach we experimented with involved using stereo image pairs to generate a full 3D point cloud of the environment and then extracting cone positions from this cloud. The idea was to take advantage of the stereo depth map to reconstruct a dense point cloud, where pixels in the image correspond to 3D points in the camera coordinate frame. In principle, this provides a geometrically accurate spatial representation of the scene.

To obtain the position of a detected cone, we attempted to locate the corresponding 3D point. This process introduced two major challenges. It required accurate pixel-level correspondence between detection outputs and point cloud entries. Due to occlusions, sparsity, and partial cone visibility, identifying the correct 3D point for each detection was unreliable and inconsistent.

Second, and more critically, the quality of the generated point cloud was highly sensitive to camera motion. During vehicle dynamics—especially under acceleration, braking, or uneven terrain—the stereo matching became unstable, leading to noisy or missing depth values.

Due to these issues—complex matching logic and poor reliability under motion—this approach was ultimately discarded. It was replaced by a more direct method that computes the 3D ray for a detected pixel and uses its associated depth value from the depth map, which offered greater robustness and simpler integration.

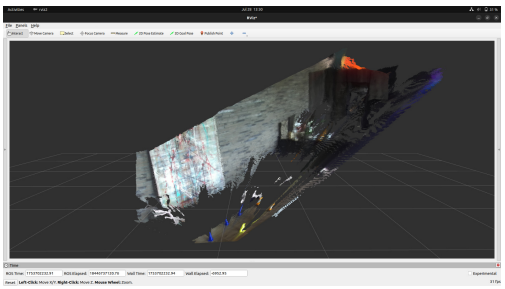

Fig. 6. Stereo Poincloud

2.3 Visual SLAM NVIDIA-accelerated

Among the visual SLAM algorithms evaluated, we explored a stereo visual-inertial odometry (SVIO) approach—cuVSLAM by NVIDIA [8]—that leverages GPU acceleration to deliver real-time, low-latency results suitable for robotics applications.

In addition to the methods previously discussed (Section 1.4), this algorithm supports the fusion of multiple stereo image pairs from different camera sources. It is also capable of incorporating IMU and GNSS data, which enhances pose estimation under challenging conditions—such as those encountered in autonomous racing scenarios. Unlike traditional VSLAM pipelines, this approach utilizes GPU resources to extract and match a larger number of keypoints in real-time, with a dedicated optimization step to minimize overall reprojection error.

While the algorithm demonstrated promising performance on the NVIDIA Jetson AGX Orin platform, we encountered several integration issues when deploying it on x86 systems. Given these limitations—and considering the satisfactory results achieved with our alternative method—we ultimately chose not to adopt this algorithm in our prototype. Further testing and on-track comparisons will be necessary to determine the most suitable SLAM solution for final deployment.

Method	Runtime	Translation	Rotation
VSLAM	0.007s	0.94%	0.0019 deg/m
ORB SLAM2	0.06s	1.15%	0.0027 deg/m

Table 2. Comparison of SLAM methods based on performance metrics. VSLAM results are sourced from NVIDIA's cuVSLAM benchmarks [8] as measured on KITTI [9] Visual Odometry / SLAM Evaluation 2012 for real-time applications.

3 CONCLUSIONS AND RESULTS

In this paper, we presented a comprehensive visual mapping pipeline tailored for autonomous racing vehicles, specifically designed to detect traffic cones and register their locations within a persistent global map. The proposed system is composed of five main components: (i) a YOLOv11-m cone detector, fine-tuned for the specific domain of Formula Student Driverless competitions; (ii) a visual-inertial SLAM module for accurate and continuous vehicle pose estimation; (iii) a stereo-based depth map generator capable of delivering dense range information in real time; (iv) a fusion layer that computes 3D cone positions from combined detection and depth data; and (v) an EKF-based mapping module that integrates all observations to maintain global map consistency over extended operation. All modules were deployed within a ROS 2 environment, ensuring efficient inter-module communication and real-time performance.

The complete pipeline was evaluated under experimental conditions representative of Formula Student Driverless tracks, including challenging factors such as high vehicle dynamics, variable illumination, and partial occlusions. Our results confirm that the YOLOv11-m detector, when combined with domain-specific training and augmentation, achieves a high level of robustness in detecting cones across diverse environments. The stereo depth estimation proved effective in providing reliable distance measurements, while the visual-inertial SLAM module maintained stable odometry even during aggressive maneuvers.

The integration of these components through the fusion and EKF layers enabled the system to build and refine a persistent, globally consistent cone map as the vehicle completed multiple laps. This approach reduced the impact of transient detection errors and odometry drift, allowing for long-term mapping without significant degradation. The modular architecture also offers flexibility: individual components can be updated or replaced without requiring major changes to the rest of the pipeline.

Overall, the experimental results demonstrated that learning-based detection, when paired with stereo geometry and probabilistic state estimation, forms a powerful combination for robust, accurate, and real-time map construction in dynamic racing environments. These findings suggest that the system is well-suited for deployment in Formula Student Driverless competitions and could be adapted for other autonomous driving contexts where lightweight, real-time perception is essential.

Future work will focus on integrating additional sensor modalities, such as LiDAR or RTK-GNSS, to further improve mapping accuracy and reliability.

Fig. 7. Bird-Eye-View

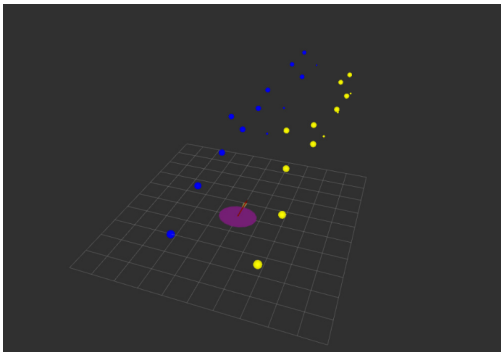

Fig. 8. Bird-Eye-View

4 ACKNOWLEDGMENTS

We would like to express our sincere gratitude to our professors for their guidance and support throughout this course. We are also thankful to our classmates and peers for being open to constructive confrontation and for their encouragement, which greatly enriched our learning experience.

5 REFERENCES

[1] ANFIA, "Formula SAE Italy (Formula ATA)," (2025), URL https://www.formula-ata.it/formula-sae-italy/, accessed: 2025-09-03.

[2] s. R. Khanam, s. M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," arXiv preprint arXiv:2410.17725 (2024), URL https://arxiv.org/abs/2410.17725, submitted 23 October 2024; accessed 3 September 2025.

[3] H. Durrant-Whyte, T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110 (2006).

[4] S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics* (MIT Press, Cambridge, MA) (2005).

[5] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, "Robot Operating System 2: Design, Architecture, and Uses In The Wild," arXiv preprint arXiv:2211.07752 (2022), URL https://arxiv.org/abs/2211.07752, submitted on 14 November 2022; accepted for publication in *Science Robotics* (Vol. 7, No. 66, 2022).

[6] D. Dodel, M. Schötz, N. Vödisch, "FSOCO: The Formula Student Objects in Context Dataset," arXiv preprint arXiv:2012.07139 (2020), available: https://arxiv.org/abs/2012.07139.

[7] Bradski, Gary and the OpenCV community, *OpenCV: Open Source Computer Vision Library*, OpenCV.org (2025), URL https://docs.opencv.org/, accessed: 2025-09-03.

[8] NVIDIA, "cuVSLAM Overview - Isaac ROS Documentation," (2024), available: https://nvidia-isaac-ros.github.io/concepts/visual_slam/cuvslam/index.html.

[9] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, "Vision meets Robotics: The KITTI Dataset," presented at the *The International Journal of Robotics Research (IJRR)*, vol. 32, pp. 1231–1237 (2013).