ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Artificial Intelligence in Industry

PUSHING CARS' LIMITS: EXPLORING AUTONOMOUS TECHNOLOGIES IN THE FORMULA SAE DRIVERLESS COMPETITION

CANDIDATE
Edoardo Fusa

SUPERVISOR
Prof. Michele Lombardi

Academic year 2024-2025

Session 1st

As shown in Figure 5.5, both the electric ASB actuator and the pneumatic EBS cylinders are integrated into a single mechanical unit. They act on the same master cylinders, which transmit pressure to the hydraulic brake calipers. The hydraulic circuits are managed by **shuttle valves**, which ensure that whichever system generates the higher pressure—be it the ASB, the EBS, or the manual driver's pedal—is the one that actuates the brakes, preventing interference between the independent inputs.

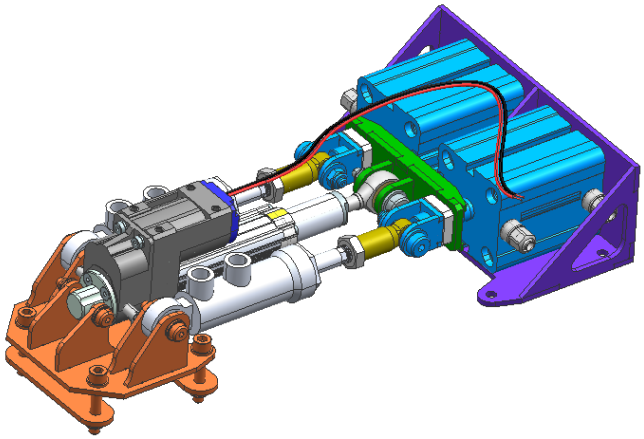

Figure 5.5: The combined assembly housing the electric Autonomous System Brake (ASB) actuator and the pneumatic Emergency Brake System (EBS) cylinders.

5.2 Stereocamera Pipeline

The perception system relies on a stereocamera to detect the track layout and reconstruct the 3D positions of the cones. This process begins with acquiring images from the sensor and concludes with a list of 3D landmarks, complete with color information, in the vehicle's coordinate system.

5.2.1 Sensor Characteristics and Operational Limits

Our system utilizes a ZED 2i stereocamera, fitted with polarizing filters to mitigate glare. The camera is configured with the following operational parameters:

- **Resolution & Framerate:** 2x (1280x720) at 60 fps
- **Focal Length:** 4 mm
- **Baseline:** 12 cm
- **Field of View (FOV):** 72° (H) x 44° (V) x 81° (D)
- **RGB Sensors:** Dual 1/3" 4MP CMOS, 2µm pixel size, rolling shutter
- **Motion Sensors:** Integrated Accelerometer and Gyroscope. The magnetometer cannot be used, as its readings are heavily compromised by interference from the vehicle's nearby metallic main hoop[1].
- **Physical Specs:** 175.3 x 30.3 x 43.1 mm, 229 g
- **Power:** 380 mA at 5V, supplied via USB-C

The manufacturer's specifications on depth perception provide critical insight into the system's operational limits. The ideal depth range is stated as 1.5 m to 20 m, with depth accuracy degrading significantly with distance. At 2 m, the accuracy is better than 0.4% (<8 mm), but at 20 m, it falls to less than 7%, which corresponds to a potential error of 1.4 meters. This is a significant margin of error that implies that the performance of our perception algorithms will inherently be less reliable for cones detected at longer distances.

[1] The main hoop is a mandatory element that each car needs to have [9].

5.2.2 Image Acquisition and Preprocessing Node

The pipeline's first stage is a custom ROS 2 node that wraps an open-source driver [15] to interface with the camera. This node is responsible for acquiring the raw image data and performing initial validation and processing.

A significant operational challenge identified during testing is the camera's default automatic exposure setting. When faced with a bright sky, the camera often underexposes the lower portion of the image, making the traffic cones difficult to discern. To address this, future work will focus on implementing manual control over the camera's exposure settings.

This node's primary responsibility, however, is to handle a data corruption issue observed in practice. At a non-negligible rate, the camera driver outputs a "shifted" frame, where the left and right images are not correctly placed side-by-side (Figure 5.6).

Figure 5.6: Example of a corrupted output from the camera driver. Instead of a single, sharp discontinuity at the center, multiple misaligned seams are present.

Processing such a frame would lead to erroneous depth calculations. To prevent this, a data validation algorithm is implemented as a sanitation check. It exploits the fact that a correctly formed stereo image has a sharp visual discontinuity along the central vertical seam. The algorithm operates as follows:

1. It isolates a narrow vertical strip at the horizontal center of the incoming stereo frame.

2. It calculates the column-wise Sum of Absolute Differences (SAD) in pixel intensity directly on the central seam, as well as on the columns

immediately to its left and right.

3. It computes a ratio of these sums to determine if the visual difference at the seam is significantly larger than in the adjacent, continuous parts of the image.

If the difference at the seam is not pronounced, the algorithm identifies the frame as corrupted, and it is subsequently discarded. Testing of this validation algorithm showed a perfect F1 score, both on-track and off-track. Once a frame is validated, it is rectified using the precise manufacturer-provided calibration matrices for our specific camera unit. The resulting pair of rectified images, along with the corresponding calibration data, are then published to the ROS local network for consumption by the downstream perception nodes.

5.2.3 Cone Detection with YOLO

Once the synchronized left and right image frames are received, the first step in the perception pipeline is to detect the traffic cones within each image. This task is performed by a custom-trained You Only Look Once (YOLO) deep neural network. The network version used is *yolov11n* [10]. To achieve the real-time performance required for autonomous racing, the inference process is accelerated using hardware-specific optimization frameworks. The system is designed to leverage either the NVIDIA TensorRT engine [29] for NVIDIA GPUs or the OpenVINO toolkit [28] for Intel hardware[2]. The selection of the framework has to be done before the pipeline compilation.

The detection process for each stereo pair follows these steps:

1. **Image Preprocessing:** The raw left and right images are first preprocessed to match the fixed input dimensions required by the YOLO model (i.e., 640x640 pixels). To avoid distorting the objects, the original aspect ratio is maintained. Each image is resized to fit within the model's

[2]OpenVINO has been chosen also as a way to debug the pipeline if only an Intel CPU was available. Though, on the ASU, TensorRT is the main framework.

input dimensions, and the remaining space is filled with black padding.

2. **Batched Inference:** A key performance optimization is the use of batched inference. The preprocessed left and right images are stacked into a single batch of size two. This batch is then fed to the inference engine (TensorRT or OpenVINO) in a single pass, which is significantly more efficient than processing the two images sequentially.

3. **Post-processing and Filtering:** The raw output from the neural network is a list of potential object detections with associated confidence scores. This output is post-processed to yield the final set of bounding boxes. First, all detections with a confidence score below a predefined threshold are discarded. Then, Non-Maximum Suppression (NMS) is applied. NMS is a crucial algorithm that eliminates redundant, overlapping bounding boxes for the same object, ensuring that only the single best bounding box for each detected cone is retained.

Figure 5.7: YOLO output after NMS. The numbers represent the confidence (or probability) of the detections.

The output of this stage consists of two independent lists of 2D bounding boxes: one for the cones detected in the left image and one for the cones detected in the right image. Each bounding box contains the pixel coordinates, dimensions, and class ID (e.g., blue cone, yellow cone) of a detected cone.

5.2.4 Stereo Matching of Detections

After detecting cones in both images, the next critical step is to solve the data association problem: determining which bounding box in the left image corresponds to which bounding box in the right image. This stereo matching process is fundamental for 3D pose estimation. Our pipeline employs a multi-stage approach to find robust and accurate correspondences.

1. **Geometric Candidate Filtering:** For each bounding box in the left image, a search is performed to find potential matching candidates in the right image. This search is heavily constrained by geometric rules derived from the stereo camera setup. A right-image bounding box is considered a valid candidate only if it satisfies several conditions relative to its left-image counterpart:

    - The bounding boxes' vertical centers must be very similar, in accordance with the epipolar constraint of a rectified stereo system.
    - The height of the bounding boxes must be comparable.
    - The horizontal center of the right-image box must be to the left of the left-image box's center, and the difference in their positions (the disparity) must fall within a plausible range. This range is dynamically adjusted based on the object's vertical position in the image, allowing for larger disparities for closer objects.
    - The algorithm includes special logic to handle cones that are partially visible at the edges of the frame, which improves robustness.

2. **Template Matching for Refinement:** If one or more candidates are found, the matching is refined using template matching. The image patch defined by the left bounding box is used as a template. This template is then searched for within a constrained horizontal region of the right image, defined by the positions of the candidate boxes. Template matching is used to find the location of the best match. To achieve

higher precision, the disparity is refined to sub-pixel accuracy by fitting a parabola to the template matching scores around the peak response.

3. **Feature-Based Correspondence:** Inspired by [11], to further improve the robustness of the 3D position estimate, an optional feature-matching step can be enabled. For each matched pair of bounding boxes, an ORB (Oriented FAST and Rotated BRIEF) [23] feature detector is used to find multiple stable keypoints within each box's image patch. These keypoints are then matched between the left and right patches. This provides a set of multiple, spatially distributed correspondence points for a single cone, which is more resilient to minor inaccuracies in the bounding box itself. If this step is disabled or fails to find matches, the system falls back to using the sub-pixel-refined center point of the bounding boxes as the single correspondence point.

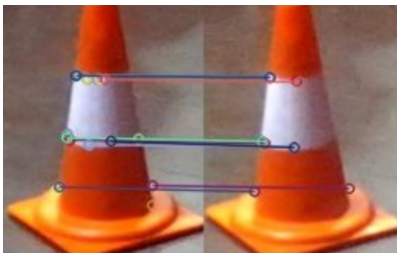

Figure 5.8: Example of ORB feature matching on a pair of left-right bounding boxes.

The result of this stage is a list of matched cone pairs. Each pair contains a set of corresponding 2D points (one or more) from the left and right images that belong to the same physical cone.

5.2.5 3D Landmark Triangulation

The final step in the pipeline is to use the 2D matched points to reconstruct the 3D position of each cone relative to the camera. This process is known as triangulation.

For each matched cone, the set of corresponding 2D keypoint pairs and the camera's pre-calibrated projection matrices are passed to the standard OpenCV [6] triangulation (*cv::triangulatePoints*) function. This function uses a linear triangulation method to compute the 3D coordinates for each keypoint pair. This results in a small 3D point cloud for each cone.

To obtain a single, robust 3D position estimate from this point cloud, the **median** of the X, Y, and Z coordinates is calculated. Using the median is a highly effective technique for rejecting outlier points that may have resulted from incorrect feature matches, leading to a more stable final position estimate[3].

Furthermore, the system also estimates the uncertainty of this 3D position. The variance of the position is calculated based on the estimated depth and the known uncertainty of the disparity measurement in pixels. This provides a covariance matrix for each landmark, which is essential information for downstream probabilistic filters, such as the SLAM system.

Finally, the calculated 3D point, now in the camera's coordinate system, is transformed into the vehicle's reference frame using the known extrinsic calibration (the position and orientation of the camera on the car's chassis). The final output of the stereocamera pipeline is a list of 3D landmarks, each with a position and an associated covariance, ready to be used for mapping and navigation.

5.2.6 Experiments and Results

Custom YOLO training

The cone detection model was trained and evaluated using the publicly available FSOCO (Formula Student Objects in Context) dataset [33]. The model was trained for 300 epochs, and its final performance was measured on a test split, which comprises 1,968 images containing a total of 36,123 annotated

[3] A likely more accurate way to do this would be to compute the geometric median, at the cost of solving an optimization problem.

cone instances. The dataset composition is shown in Table 5.1, while the performance metrics are summarized in Table 5.2.

The primary metrics are defined as follows:

- **Precision:** Measures the accuracy of the detections. Of all the predictions made by the model, this is the fraction that were correct. A high precision indicates a low rate of false positives.
- **Recall:** Measures the model's ability to find all relevant objects. Of all the actual cones present in the images, this is the fraction that the model successfully detected. A high recall indicates a low rate of false negatives.
- **mAP50:** The mean Average Precision calculated at an Intersection over Union (IoU) threshold of 0.50. This metric evaluates how well the model can both classify an object and localize it with a bounding box that overlaps at least 50% with the ground truth box.
- **mAP50-95:** The mAP averaged over a range of IoU thresholds from 0.50 to 0.95. This is a much stricter metric, as it requires highly accurate bounding box placement to score well.

As shown in the *All Classes* summary row, the model achieves an overall mAP50 of 0.824, indicating strong general performance. The precision of 0.849 suggests that when the model detects a cone, it is correct about 85% of the time, while the recall of 0.765 means it successfully identifies roughly 77% of all cones present in the dataset.

The per-class breakdown reveals further insights. The model performs exceptionally well on the primary cone classes: *Blue*, *Yellow*, and *Orange*. The precision for these classes is very high (above 0.92), meaning there are very few incorrect classifications. The mAP50 scores for these classes are also excellent, nearing 0.90, which confirms the model's ability to reliably detect and localize the main track boundary markers.

The *Large Orange Cone* class achieves the highest mAP scores (0.908 for mAP50 and 0.710 for mAP50-95). This is likely because these cones are larger and more visually distinct, making them an easier target for the detector.

Conversely, the *Unknown Cone* class exhibits significantly lower performance across all metrics (mAP50 of 0.547). This result is expected, as this class serves as a catch-all for heavily occluded, blurry, or poorly illuminated cones that are difficult to classify even for a human annotator. The model's struggle with this ambiguous class is acceptable, as its primary function is to accurately identify the well-defined track boundaries. Overall, the results confirm that the custom-trained model is highly effective for its intended purpose.

Table 5.1: Composition of the FSOCO test split used for model evaluation.

Class	**Images**	**Instances**
All Classes	1968	36123
Blue Cone	1416	12720
Yellow Cone	1638	15605
Orange Cone	850	5462
Large Orange Cone	428	1263
Unknown Cone	178	1073

Table 5.2: Performance metrics of the trained cone detector on the FSOCO test split.

Class	**Precision**	**Recall**	**mAP50**	**mAP50-95**
All Classes	0.849	0.765	0.824	0.570
Blue Cone	0.922	0.806	0.896	0.615
Yellow Cone	0.926	0.794	0.892	0.608
Orange Cone	0.925	0.787	0.877	0.603
Large Orange Cone	0.868	0.873	0.908	0.710
Unknown Cone	0.603	0.566	0.547	0.315

Performance and Timing Analysis

To evaluate the real-world performance of the stereocamera pipeline, its timing characteristics were benchmarked using data recorded during a manually

driven lap at the Rioveggio Karting Circuit. The test was executed on a high-performance machine with hardware comparable to the vehicle's ASU, featuring an AMD Ryzen 9 6900HX CPU and an NVIDIA GeForce RTX 3080 Mobile GPU.

The pipeline was benchmarked across four distinct configurations to assess the impact of both the inference engine (TensorRT vs. OpenVINO) and the feature matching strategy (sub-pixel refined center point vs. ORB features). The timing statistics, averaged over 1870 frames, are presented in Table 5.3 and Table 5.4.

Table 5.3: Mean and standard deviation of processing times (in milliseconds) for the TensorRT configuration, running on an NVIDIA RTX 3080 GPU. In parenthesis the standard deviation.

Processing Stage	**Center Point**	**ORB Features**
Preprocessing	0.34 (0.08)	0.34 (0.07)
Inference	6.78 (2.45)	6.78 (2.05)
Postprocessing	0.38 (0.08)	0.38 (0.08)
BBox Matching	0.38 (0.27)	0.40 (0.28)
Feature Matching	0.00 (0.00)	1.41 (4.27)
Triangulation	0.02 (0.01)	0.11 (0.07)
Sending	0.04 (0.02)	0.04 (0.01)
Total	**7.95 (2.57)**	**9.46 (6.35)**

Table 5.4: Mean and standard deviation of processing times (in milliseconds) for the OpenVINO configuration, running on an AMD Ryzen 9 CPU. In parenthesis the standard deviation.

Processing Stage	**Center Point**	**ORB Features**
Preprocessing	1.31 (0.19)	0.37 (0.08)
Inference	24.72 (1.61)	24.30 (1.25)
Postprocessing	0.59 (0.07)	0.60 (0.06)
BBox Matching	0.50 (0.36)	0.49 (0.35)
Feature Matching	0.00 (0.00)	1.76 (2.69)
Triangulation	0.03 (0.01)	0.15 (0.08)
Sending	0.05 (0.01)	0.05 (0.01)
Total	**27.20 (1.74)**	**27.72 (3.50)**

The results highlight several key aspects of the pipeline's performance:

- **Inference Engine Impact:** The most significant factor influencing total processing time is the inference engine. The GPU-accelerated TensorRT configuration is approximately **3.6 times faster** at the core inference task than the CPU-based OpenVINO configuration (6.78 ms vs. 24.72 ms). This demonstrates the critical importance of GPU hardware acceleration for running deep learning models in a real-time context.

- **Feature Matching Overhead:** The choice of matching strategy introduces a clear trade-off between performance and robustness. Using ORB features adds a noticeable overhead, increasing the total processing time by about 1.41 ms in the TensorRT case and 1.76 ms in the OpenVINO case. The *Feature Matching* and *Triangulation* stages take significantly longer, as they must process multiple keypoints per cone instead of a single center point. This additional processing time is the cost of achieving a potentially more stable 3D position estimate by relying on multiple geometric correspondences. It is also worth noting the high standard deviation of the feature matching stage, which is expected as the computation time is directly proportional to the number of times ORB is executed[4], a value that varies significantly from frame to frame.

- **Real-Time Capability:** All four configurations operate well within the time budget required for our 60 fps camera, which is approximately 16.7 ms per frame. The TensorRT configurations are fast, with the full pipeline completing in under 10 ms. This leaves a margin (even though not too comfortable) for all other processes running on the ASU, such as SLAM and MPC. The OpenVINO configuration, while slower, still operates at an acceptable speed (27 ms, or 37 Hz), making it a viable fallback option if dedicated GPU hardware is unavailable.

[4]ORB is executed sequentially for each pair of bounding boxes. Future work could focus on executing ORB on GPU if it is a non-negligible advantange.

3D Position Estimation Validation

To validate the 3D accuracy of the perception pipeline, a series of static tests were conducted outdoors on an asphalt surface. With the stereocamera fixed in place, a standard yellow competition cone was positioned at four known ground truth distances along the camera's forward-facing Z-axis: 5 m, 10 m, 15 m, and 20 m. For each position, 17 consecutive measurements were recorded to evaluate the stability and accuracy of the depth estimation for both the center point and ORB features methods.

The results of this experiment, showing the mean and standard deviation of the estimated Z-distance, are summarized in Table 5.5.

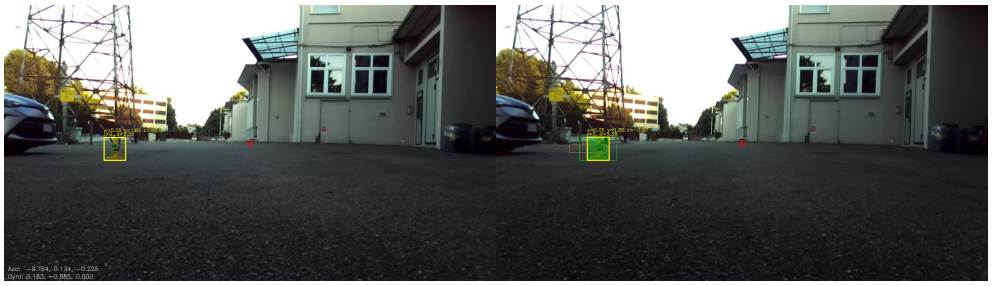

Figure 5.9: Yellow cone placed at 5 m in the experiment.

Figure 5.10: Yellow cone placed at 10 m in the experiment.

The analysis of these results provides several important insights into the pipeline's real-world performance:

- **Accuracy and Precision Decrease with Distance:** For both methods, the accuracy (how close the mean is to the ground truth) and the precision (the consistency of the measurements, indicated by the standard

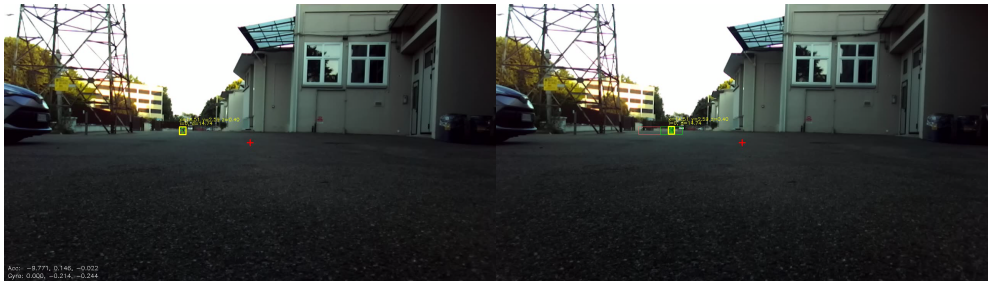

Figure 5.11: Yellow cone placed at 15 m in the experiment.

Figure 5.12: Yellow cone placed at 20 m in the experiment.

deviation) degrade as the distance to the cone increases. At 5 m, the estimates are highly accurate and stable. However, at 20 m, the mean estimation error approaches 2 meters, and the standard deviation exceeds 1 meter, indicating significant uncertainty and noise in the predictions.

- **Consistency of ORB vs. Center Point:** The key difference between the two methods lies in their consistency. At every tested distance, the standard deviation of the ORB-based method is significantly lower than that of the center point method. For instance, at 15 m, the ORB method's standard deviation (0.68 m) is nearly half that of the center point method (1.16 m). This demonstrates that using multiple feature points and taking the median provides a more stable and robust estimate, as it is less susceptible to noise and minor errors in the bounding box detection.

- **Confirmation of Sensor Limits:** The observed errors are consistent with the camera manufacturer's specifications, which state a depth accuracy of less than 7% at 20 m (an error of up to 1.4 m). Our results,

Table 5.5: Mean and standard deviation (in meters) of the estimated Z-distance for a cone placed at known ground truth distances. Results are shown for both the Center Point and ORB feature matching methods.

Ground Truth	Estimated Z-Distance: Mean (Std. Dev.)	
Distance	Center Point	ORB
5 m	5.07 (0.19)	5.08 (0.11)
10 m	9.56 (0.46)	9.87 (0.28)
15 m	14.25 (1.16)	13.95 (0.68)
20 m	18.04 (1.37)	18.35 (1.13)

with mean errors around 1.7-2.0 m at 20 m, align with this physical limitation. This confirms that for long-range perception tasks, the inherent accuracy of the sensor itself is a primary constraint on the system's performance.

In conclusion, while both methods provide comparable accuracy on average, the ORB feature-based approach offers a clear advantage in terms of measurement stability and reliability, especially at medium to long distances.

5.3 The Stack's Vehicle Dynamics Model

An accurate vehicle dynamics model is essential for developing both simulation environments and advanced control algorithms like Model Predictive Control (MPC). This section presents the single-track (or *bicycle*) model used in our software stack. The model is heavily inspired by the work of ETH Zurich's AMZ racing team [31, 25] but has been adapted to fit the specific characteristics of our vehicle, Athena.

5.3.1 Model Formulation

The model describes the vehicle's motion using a state-space representation. It operates in a vehicle-centric coordinate frame, with the X-axis pointing forward, the Y-axis to the left, and the Z-axis upward, originating at the vehicle's