Real Time and Embedded Systems Project: F1Trail

Team Members

- Pietro Moriello: Vision Nodes & ROS
- Lorenzo Sirotti: Clustering Node
- Matteo Gianferrari: Tracker Node & Team Leader
- Matteo Baccilieri: Controller Node & Simulator

Project Objectives and Requirements

- ***Main objective***: Make an F1Tenth move autonomously following a desired target with an ArUco marker attached to it.

- ***System requirements***:
    1. The car must track a marked object (ArUco marker).
    2. The car must make use of both LIDAR and vision during operation.
    3. The car must follow the tracked object maintaining a target distance (tunable hyperparameter).
    4. The car must be able to follow the target along a longitudinal direction (forwards and backwards).

Vision Nodes - sl_stereo_proc

This node serves as a bridge between the images produced by the camera that we employed (StereoLabs' Zed) and the target localisation node.

- ***Input***: stereo image produced by Zed camera (i.e. published on */zed/zed_node/stereo/image_rect_color*).
- ***Output***: split images for each stereo camera, published on different topics.

Thanks to this node, our localisation node is agnostic of the particular implementation of stereo images.

Vision Nodes - aruco_loc

- ***Input***: stereo images containing a ArUco marker (a specific marker can be selected as target through a node parameter)
- ***Output***: 3D point relative to the *camera_link* frame of reference, which corresponds to the left camera frame of reference.

This node leverages **OpenCV**'s implementation of ArUco localisation. OpenCV data types are converted to ROS2 messages thanks to the **cv_bridge** package.

Vision Nodes - static_transform_publisher

Since other nodes may have different reference frame, we decided to elect a common reference frame, named *base_link*, for which we provide transformations.

Leveraging ROS2 library ***tf2***, we create a transform from *base_link* to *camera_link* and publish it for other nodes to use. This transformation is static, meaning it is published only once when the publisher is started and the it is cached in the **tf_static** topic (created by ROS automatically).

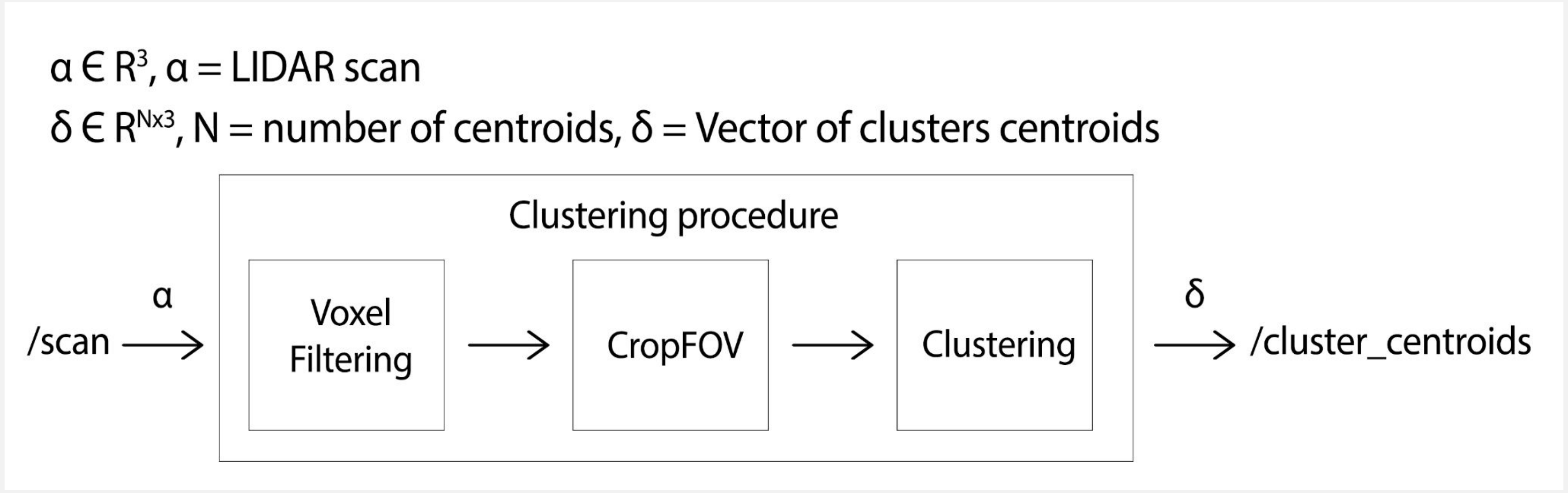
Clustering Node

This node processes a LIDAR scan into a 3D point cloud, identifies clusters and generates a vector of centroids, which is needed by the tracking node.

- ***Input***: LIDAR scan
- ***Output***: Vector of cluster centroids

Intermediate steps:

- Voxel Filtering
- CropFOV
- Clustering



Clustering Node - static_transform_publisher

Similarly to the ***aruco_loc*** node, we need to publish the transformation from *base_link* to *lidar_link*, in order to interpret data produced by this node.

Tracking Node - tracking_proc

This node receives messages containing a target location and a vector of cluster centroids, performs sensor fusion to identify the object of interest, and tracks it over time using a Kalman Filter to predict its future state.

- ***Inputs***:
  1. 3D target location (α).
  2. Vector of centroids (**δ**).
- ***Output***: 3D point representing the tracked object position (□).

Sensor fusion:

$$\alpha,\ \beta \in R^3,\ \ \delta = \{\delta_1, \ldots, \delta_N\},\ \ \delta \in R^{N\times 3}$$

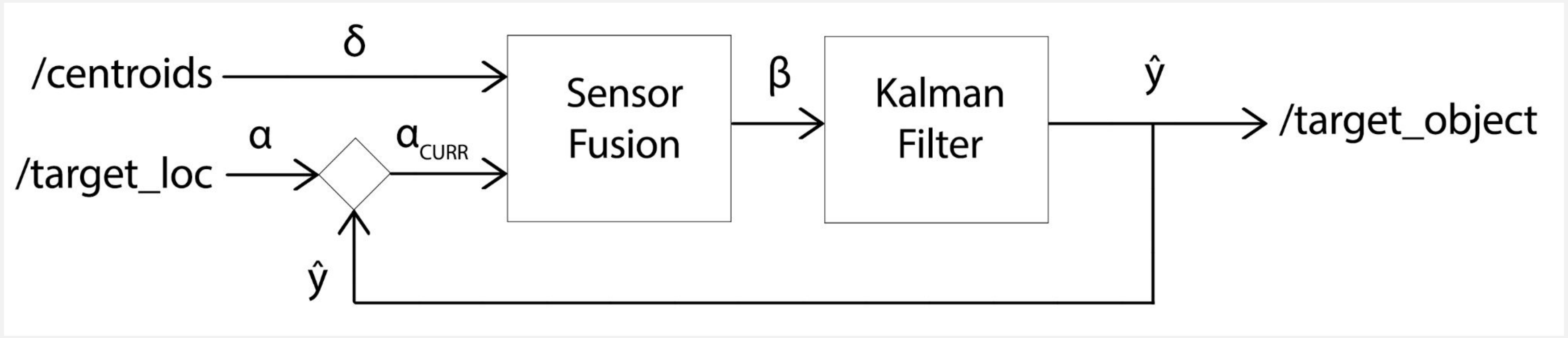
$$\beta = arg\,min_{\delta_i \in \delta} ||\alpha - \delta_i||_2$$

Tracking Node - tracking_proc

Start tracking conditions:

- The incoming vision and clustering messages can be transformed using tf2.
- 1 message from the vision and 1 message from the clustering nodes.

Tracking procedure:



Kalman Filter

Discrete-time, Linear Kalman Filter for a constant velocity model.

State vector: $x = \{p_x, p_y, p_z, v_x, v_y, v_z\},\ x \in R^6$

The measurement matrix H is equal for both Lidar and camera, both sensors output a position.

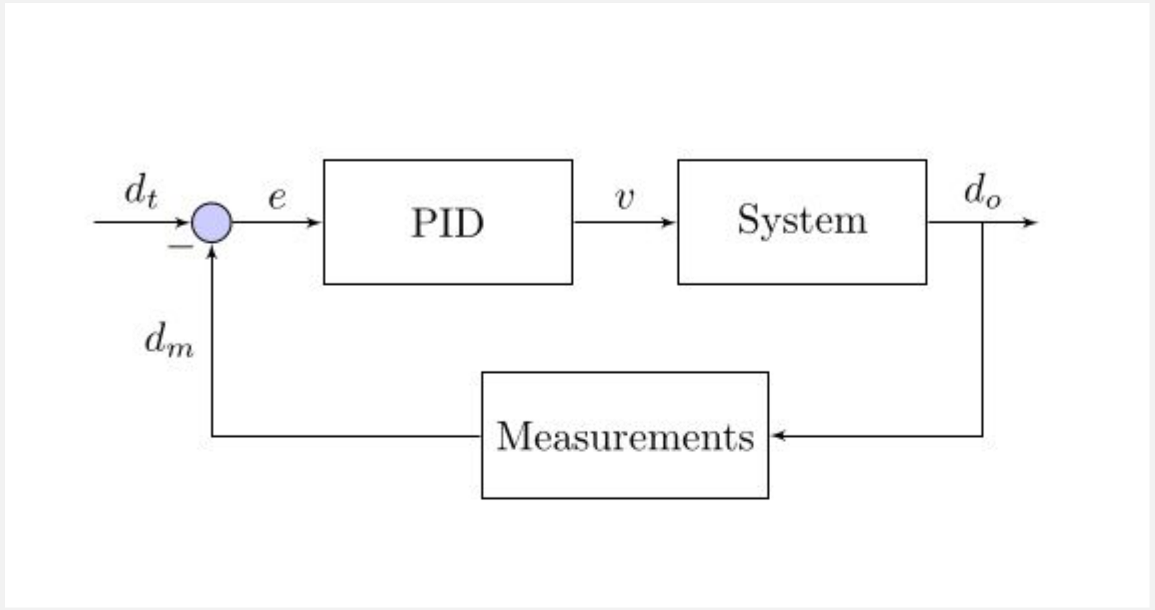
Camera and Lidar measurement covariance matrices:

$$R_{camera} = \begin{bmatrix} 0.0049 & 0. & 0. \\ 0. & 0.0049 & 0. \\ 0. & 0. & 0. \end{bmatrix} \qquad R_{lidar} = \begin{bmatrix} 0.0016 & 0. & 0. \\ 0. & 0.0016 & 0. \\ 0. & 0. & 0. \end{bmatrix}$$

The update matrix F and the process covariance matrix Q are updated based on the delta time between received measurements.

Controller Node (PID)

- ***Input***: Tracked object position and car odometry
- ***Output***: Throttle control for adjusting the speed of a vehicle to maintain a target distance



Simulation

We used the f1tenth_gym_ros to simulate an F1TENTH car within a ROS2-compatible environment. The simulation was customized to incorporate additional ROS topics, enabling real-time visibility into tracking results and improved performance analysis of autonomous algorithms.

Project limitations

Since we were able to perform tests in a static environment only, the car is not guaranteed to function as expected in dynamic environments.

Main limitations:

- Environment mapping (ex.: SLAM, particle filter for localization).
- Obstacle avoidance.
- Path planner.
- Lateral movement.
- Limited speed of the object and distance from the person.
- 3D lidar for point cloud classification with ML/DL algorithm.

Follow up

Here are some **follow-up points** for the F1Trail project:

- Code parallelization to increase performance (profiling, analysis, CPU & GPU & FPGA).
- Implementing components to overcome current limitations.
- Cleaner implementation of ROS components (lifecycle nodes, composable nodes, zero-copy messages).
- Testing, testing, testing.