Formation following: multi-sensor object detection and localization using ROS2 and Gazebo

Beatrice van Eden[2*], *Thabisa* Maweni[1*], *Tiro* Setati[1], *Thegaran* Naidoo[1*] and *John* Dickens[1*]

[1]CSIR, Centre for Robotics and Future Production, South Africa
[2]CSIR, Next Generation Enterprises and Institution, South Africa

Abstract. Formation control is essential in multi-robot systems, enabling efficient navigation, coordination, and task execution in dynamic environments. This work presents a vision-based approach for formation following using three robots. Each robot is identified by a unique colour and recognised using YOLOv8. The recognition output is combined with LiDAR distance measurements to facilitate waypoint navigation and formation maintenance. A set of hardcoded rules ensures proper formation control. The proposed method is tested in a structured environment, demonstrating effective multi-robot coordination.

1 Introduction

Multi-robot systems have emerged as promising solutions for a wide range of applications, including logistics, surveillance, and search-and-rescue operations. Multi-robot coordination can enable teams of robots to tackle complex tasks in dynamic environments. In some sense, it is like having a distributed version of a sensing and actuation system.

A key component of multi-robot coordination is formation control, which is the ability of a team of robots to move in coordinated patterns to enhance efficiency and adaptability. Various approaches to formation control exist, such as behaviour-based, virtual structure, and leader-follower methods. [1] [2] [3].

Multi-robot coordination and formation control depends on each robot having sufficient information about the other robots' locations. In a fully networked system with open communications, it is relatively easy for the robots in a multi-robot system to share this information amongst themselves. However, if the application requires that the communications between robots be restricted to a minimum, then each robot would need to obtain the information about the other robots' positions by some alternate means. A key requirement for this project is to minimise inter-robot communications while ensuring that the robots can navigate through the environment and maintain a specified formation.

For this paper, a leader-follower strategy is employed that integrates computer vision and LiDAR-based navigation to allow each follower robot to determine the location of the robot

* Corresponding author: bveden@csir.co.za

that it is to follow. In this system, the leader performs global navigation using a waypoint-based navigation system, while the follower robots rely on localized navigation by making use of computer vision and LiDAR data to find the particular robot that it needs to follow.

In this approach, each robot uses YOLOv8 [6] to identify a designated, unique robot within the formation. The results of the vision-based detection are then fused with LiDAR measurements to compute the inter-robot distances and relative positions accurately. The followers continuously adjust their positions based on the dynamic location of the lead robot, while also incorporating an obstacle avoidance navigation stack to traverse environments populated with both stationary and dynamic obstacles safely. This integrated methodology minimises inter-robot data exchange and provides a robust framework for coordinated multi-robot navigation in complex scenarios.

The main contributions of this paper are : (i) the development of a leader–follower formation strategy that minimises inter-robot communication, (ii) the integration of computer vision and LiDAR through a sensor fusion approach to achieve accurate inter-robot positioning, and (iii) the experimental validation of the system in formation control navigation with obstacle avoidance in a simulated environment.

2 Related work

Beyond vision, some research employs radio frequency identification (RFID) based methods, demonstrating the ability to continuously measure and adjust the follower robot's movement to the target's motion in real time [4]. While RFID-based object-following approaches have been explored in various applications, they often require physically tagging the target robot. This requirement limits their ability in sensitive or security applications where minimal data sharing between the target and the follower agents is critical. In contrast, vision-based multi-sensor methods like the one proposed in this paper allow formation without inter-robot data exchange.

Recent research in multi-robot systems has shown advances using vision-based methods, [5], [6] , [7] to coordinate the motion of multi-robot systems. Montijalo et al. proposed a distributed vision-based formation control framework without an external positioning system [8].Using onboard cameras and inertial measurement units, their approach relies on structure-from-motion techniques based on shared environmental features to estimate relative poses between the robots. Although demonstrated on quadrotors, this work highlights purely vision-based formation control challenges regarding robustness.

In contrast, the approach presented here employs multi-sensor fusion to estimate the relative positions, thus mitigating ambiguity problems inherent to vision-only systems. Similar vision-based leader-follower works have demonstrated estimated position errors as low as 2 % [7]. More recent research addresses the limited field of view (FOV), restricting active detection to enhance formation control performance [9]. These developments demonstrate the growing potential of vision-based methods.

3 Methodology

This multi-robot system consists of three autonomous mobile robots simulated in a structured and populated virtual environment. The robots can be configured in multiple formation configurations, which can be changed depending on the task requirements or environmental constraints.

A requirement for this system is for the robots to maintain minimal data transfer between themselves, narrowing how the robots maintain their formations and how they traverse through the environment autonomously.

3.1 Robot architecture

The work conducted in this paper is based on the HERMES Security Quad robot, depicted in Fig 1 [10]. It is a medium-sized, battery-powered autonomous vehicle designed for robust navigation and perception.

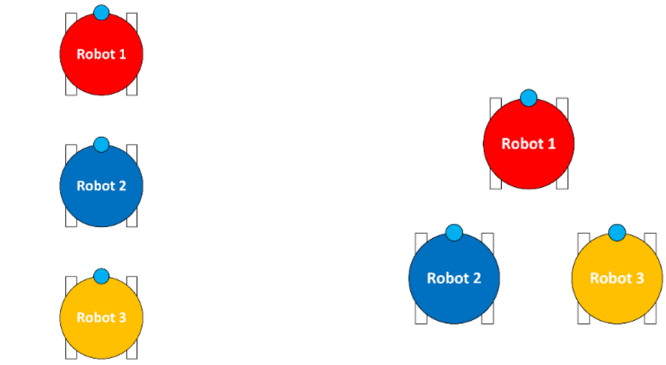
It is equipped with sensors that enable its navigation and perception capabilities, such as a GPS Receiver and RTK GPS system, an Inertial Measurement Unit (IMU), a LiDAR sensor, and a multi-camera sensor box housing four cameras on a 1.09m pole above the vehicle, giving it a 360-degree field of view.

Fig 1: HERMES robot

3.2 Leader-follower configuration

For an autonomous multi-robot system that can traverse through an environment in a desired formation, leader-follower strategy needs to be established. This strategy divides the robots into two groups: a leader robot and follower robots.

The leader robot will be responsible for establishing the path to be travelled and serve as the anchor for the formation control. The follower robots will respond to the leader by following the leader from specific distances and positions, enabling the formations to be established. The formations they can travel in are the line formation or delta formation, depicted in Fig 2and Fig 2b, respectively.



a) Line formation b) Delta formation

Fig 2: Formation configurations

3.3 Navigation

Each robot in the system will be capable of navigation and autonomously traversing through an environment as well as detect other robots by identifying and determining distances to other robots. They will also be capable of adaptive coordination, by being able to manoeuvre through and determine traversable regions in the environment using costmap-based localization; as well as formation maintenance, by using control algorithms to determine where each robot should be positioned relative to the other robots.

3.3.1 Global navigation

In this system, the leader robot will be using global navigation, allowing for the use of a simulated Global Positioning System (GPS). This GPS and costmap-based localization are used for waypoint-based path planning and trajectory execution, to manoeuvre through an environment using GPS points while dynamically avoiding obstacles encountered in the environment.

3.3.2 Local navigation

The follower robots will make use of local navigation. Because of the multi-robot system's requirement to maintain minimal data transfer, the leader robot's GPS position is not made available to the follower robots to monitor and follow. With this restriction, localized information available, such as sensor data, must be used to track and follow the leader robot while traversing through the environment.

3.4 Sensor fusion

Because local navigation necessitates the use of the robot's sensors to navigate, sensor fusion is needed. This is the fusing of the data collected from the multiple sensors on the robot, such as the LiDAR and the cameras, so the robot can have awareness of its environment, be able to navigate through the environment in real time and detect objects of interest as it travels.

Sensor fusion can help with object detection, determining distances to objects of interest, and computing the position of an object of interest relative to the follower robot.

3.5 Formation control

Once a follower robot has successfully determined the position of the leader robot relative to itself, it must move to its required position in the desired formation. This is achieved by the follower robot dynamically updating its target goal position and orientation and planning a path to that position. This is repeated at each control cycle as the follower robot maintains its position in the formation.

While the follower robot maintains its position in the formation, it must avoid obstacles in the environment. The follower robot must then determine what is the best path to take to maintain its place in a formation while avoiding obstacles in the environment.

4 Implementation

The multi-robot system is built on a custom ROS2 Humble–based software stack, with a mix of Python and C++ ROS2 nodes running on Ubuntu 22.04 LTS. The core components include

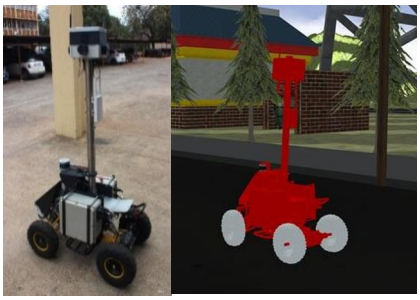
perception, navigation, sensor fusion, and formation-control nodes, all orchestrated via ROS2's middleware and launch system.

The existing HERMES robot shown in Figure 1 was simulated within the Gazebo environment and mirrors the physical robot's sensor suite and configuration.

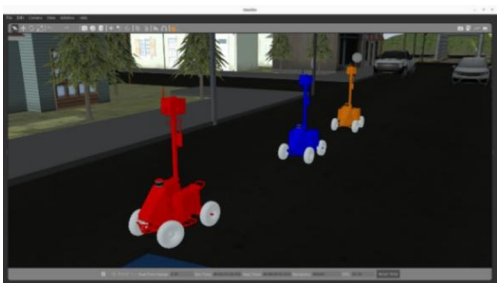
4.1 Simulation setup

Gazebo is an open-source robotics simulator that provides high-fidelity physics, sensor modelling, and environment interaction, integrated with ROS. It supports a wide range of simulated sensors (e.g., LiDAR, cameras, IMUs, GPS) and vehicle dynamics, making it ideal for developing and validating autonomous navigation, perception, and control algorithms in a virtual setting before deploying to real hardware.

Gazebo was used to model the platform by importing CAD drawings of the HERMES robot to build a URDF model (Figure 3a). Three HERMES robot models, each with a unique colour scheme for camera-based identification, can be seen in Fig 4b.



a) Real world HERMES robot (left) and Gazebo simulation of HERMES (right)

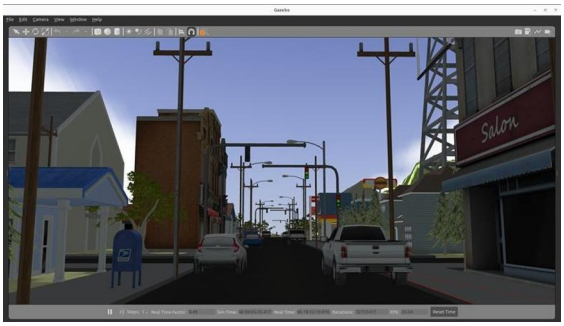


b) Leader robot (left) and two follower robots (middle and right)

Fig 3: HERMES robots

All algorithm development and testing were performed in the simulated urban world, "City World", shown in Figure 4a and Figure 4b. The "City World" features urban roads, buildings, and static and dynamic obstacles.

a) Top view of the City World simulation with the three robots



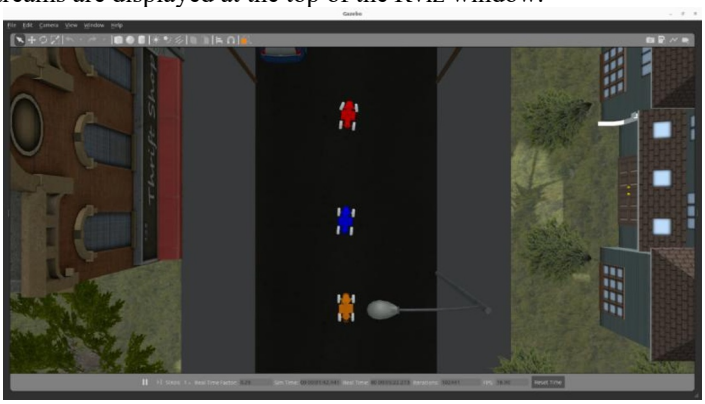
b) Ground view of the City World environment

Fig 4: Gazebo **s**imulation environment

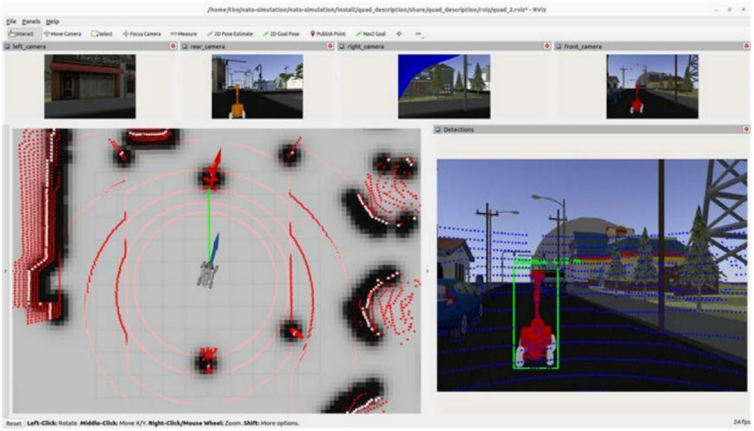
RViz is a three-dimensional (3D) visualization tool in the ROS ecosystem that displays robot states, sensor streams, and coordinate transforms, making it useful for debugging and monitoring. RViz can tap into real-world sensors or simulated sensors within the Gazebo environment using the same interface. This means that RViz can also be used in the same way as the real-world HERMES robot. In this project, RViz is used in the following ways:

- **Visualisation of sensor data**
  RViz renders and visualizes the robot's camera live feed and LiDAR point clouds.
- **Inspecting path plans**
  RViz shows both global and local trajectories as the robots plan and navigate the environment using Nav2.
- **Debugging transformations**
  This is used to verify that all coordinate frames are correctly aligned with the robot.
- **Monitoring of robot state**
  This is achieved RViz tracking the robot's odometry, goal poses, and detection annotations in real-time.

Figure 5 shows the simulation's Gazebo and RViz perspectives. The left, rear, right, and front camera streams are displayed at the top of the Rviz window.



(a) Top view of Gazebo simulation of City World environment with three robots



(b) RViz dashboard from the point of view of *Quad 2* (middle robot in (a))

Fig 5: Gazebo and RViz visualisations

4.2 Object detection and distance estimation

In this system, each follower robot runs a ROS2 node encapsulating YOLOv8, processing all its camera streams to detect the uniquely coloured leader robot or preceding robot in the formation. The resulting bounding box coordinates and confidence scores are then paired with LiDAR range data in the sensor-fusion module to compute the target robot's position and to derive the follower's required offset.

4.2.1 Dataset generation

To train the YOLOv8 detector on the HERMES robot, a custom dataset was created using a three-stage dataset creation pipeline:

1. **Capturing raw image data**
   Each robot was captured by recording ROS2 bag files of onboard camera topics with varied positions (heights: 0.5 m - 0.9 m, distances: 1.0 m - 2.0 m and yaw: $\pm\pi/6$). Camera poses were adjusted using the *.xacro* model files, and redundant or impractical views were excluded.
2. **Frame extraction**
   A custom ROS2 node extracted individual frames during bag file playback based on coverage criteria.
3. **Annotation via Roboflow**
   Selected frames were uploaded to Roboflow, a cloud-based dataset management platform. The dataset was manually labelled and exported in YOLOv8 format. Figure 5 is a snapshot of the annotations and the dataset split (train: 70 %, validate: 20 %, test: 10 %).

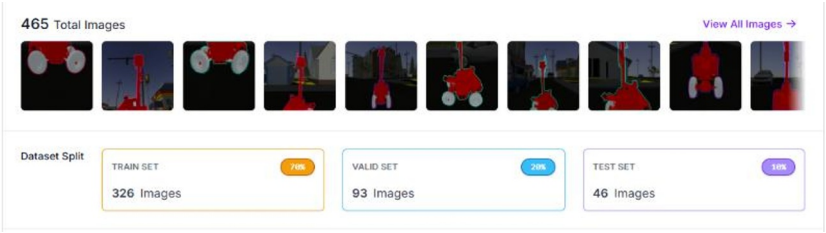


Fig 6: Dataset preparation from left, right, and rear of the robot.

4.2.2 Model training

Model training was performed in Google Colab, a cloud-hosted service with access to high-performance Graphics Processing Unit (GPU) acceleration. Data augmentation (e.g., random flips, rotations, and colour jitter) was applied to improve model generalisation, and the YOLOv8 model was trained for 100 epochs using a batch size of 16 and a 640 x 640 input image size. Training progression was tracked using loss curves and mean Average Precision (mAP) metrics on a held-out validation set. Hyperparameters were fine-tuned to optimise detection performance based on observed convergence and accuracy trends.

4.2.3 Algorithm integration

The YOLOv8 detector was encapsulated within a ROS2 node written in Python. The YOLOv8 algorithm was used via the Ultralytics package. Upon startup, the yolo detection node initializes the pretrained model weights and subscribes to the follower robot's camera image topic. For each incoming frame, it calls the YOLOv8 function and produces bounding boxes and confidence scores for detected robots in the images. The node also synchronises with the LiDAR driver by listening to scan messages. Time-stamp matching ensures that each detection can be paired with the corresponding range data in the data fusion module (Section 4.4). The deployment of the node uses a standard ROS2 launch file that configures node. In this way the node is integrated into the broader robot operating system.

4.3 Data fusion and distance estimation

Robust localisation of the target robot requires combining computer vision detections with LiDAR range measurements. The following steps describe the implementation of the fusion pipeline.

4.3.1 Input data

Two data sources feed into the data fusion module:

- **Camera images**
  RGB frames published by the follower's on-board cameras, calibrated to provide intrinsic parameters (focal lengths f_x,f_y and principal point offsets c_x, c_y).
- **LiDAR point clouds**
  3D point clouds *(x,y,z)* published in the LiDAR's coordinate frame.

4.3.2 Camera detection path

The camera images are passed to the YOLOv8 node, which outputs a set of detections. Each detection includes:

- Bounding box (x_1, y_1, x_2, y_2)
- Class label of the target robot
- Confidence score of the detection

These bounding boxes define Regions of Interest (ROIs) onto which the LiDAR points are mapped.

4.3.3 LiDAR preparation

The LiDAR points are transformed into the camera coordinate frame using the extrinsic calibration matrix $T_{\text{lidar}\rightarrow\text{camera}}$:

$$P_{\text{camera}} = T_{\text{lidar}\rightarrow\text{camera}}\, P_{\text{lidar}} \quad (1)$$

This alignment between the LiDAR and camera is essential for accurate projection of 3D points onto the 2D image plane.

4.3.4 Point projection and regions of interest (ROI) association

Each transformed LiDAR point (X, Y, Z) is projected into pixel coordinates (u, v) via the pinhole camera model:

$$u = f_x \frac{X}{Z} + c_x,\ v = f_y \frac{Y}{Z} + c_y \quad (2)$$

Points whose (u, v) fall inside a YOLO bounding box are candidates. When multiple points lie within the ROI, the one with the smallest depth (Z) (i.e., closest to the camera) is selected to represent the distance to the detected robot.

The point projection and ROI of the follower robot looking toward the leader robot is depicted in Figure 7. The numbered labels represent the following information:

1. A representation of the follower robot observing its leader.
2. The leader robot viewed from the follower robot's front camera.
3. Detection bounding box around the leader robot. This is placed by YOLOv8 running on the follower robot upon detection of the leader robot in the camera view.
4. The LiDAR points associated with the leader robot. LiDAR points that fall within the bounding box in Point 3 are projected in white. The LiDAR points not associated with the leader robot remain red.
5. The distance to the leader robot. From Point 4, the closest LiDAR point to the follower robot Point 1 is used to determine this distance.

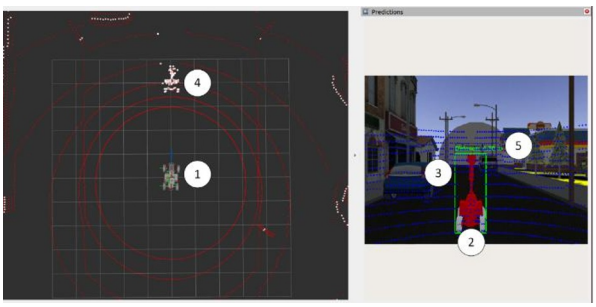


Fig 7: RViz visualisation of the point projection and ROI in the simulation

4.3.5 Position estimation

The chosen LiDAR point provides a 3D coordinate (X, Y, Z) of the target in the camera frame. This coordinate is then converted to the follower's 'base_link' frame using standard ROS2 transform tools. This gives a relative position $(x_{rel}, y_{rel}, z_{rel})$.

4.3.6 ROS message conversion

The relative position is packaged into a 'geometry_msgs/PointStamped' message (including timestamp and 'base_link' frame ID). For navigation purposes, it is then converted into a 'geometry_msgs/PoseStamped' message, adding a default orientation (quaternion) to form a valid goal pose. This pose is then published to the follower robot's navigation stack as the next waypoint, enabling the robot to adjust its trajectory and maintain the formation.

This pipeline allows the system to achieve localization of the leader robot given the communication constraints.

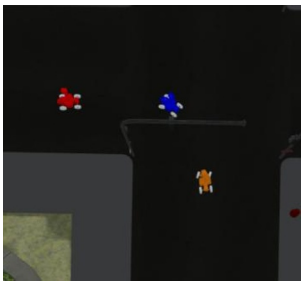
4.4 Multi-robot coordination and formation control

To enable coordination with minimal data exchange, a dedicated formation controller is required which maintains each follower's position relative to the leader. The leader traverses a predefined path, while followers detect and estimate the leader's pose, compute their own goal positions based on formation offsets, and navigate accordingly.
A custom ROS2 package, *quad_formation*, was created to implement this functionality:

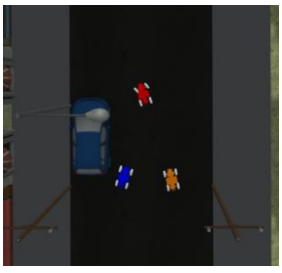
- The formation controller node executes dedicated formation control nodes for the following robots *Quad 2* and *Quad 3*.
- The *formation_controller* nodes define the position of each robot in the formation relative to the lead robot and place goal poses for the robot to follow.

4.4.1 Formations

In a line formation, shown in Figure 8a, all three robots follow one another along a single line. *Quad 2* maintains a 2 m gap directly behind the leader; *Quad 3* then follows *Quad 2* at the same 2 m separation. While in a delta formation, shown in Figure 8b, the two followers are positioned symmetrically behind the leader but offset laterally. *Quad 2* is placed 2m behind and 2 m to the right of the leader, while *Quad 3* is 2 m behind and 1m to the left.



a) Gazebo simulation with all three robots in formation



b) Gazebo simulation with all three robots in formation

Fig 8: Simulated line and delta formations

4.4.2 Implementing the formation controller

The implemented YOLO recognition node determines the direction and distance of lead robot in the follower's camera frame. In the YOLO node, these values are transformed into coordinates in the robot's 'base_link' frame to get the position of the leader relative to the follower robot as a PointStamped message on the '/leader_point' topic. Each robot's formation controller node subscribes to the topic on which the *PointStamped* message from its YOLO detection node is published.

The controller performs the following functions:

- Each formation defined has its own configuration file, where the offset position of each follower, relative to the lead robot, is defined.
- Each follower robot's position offsets are passed to its *formation_control* node.
- The *PointStamped* message from the YOLO detection node is read into the *formation_control* node.
- The data from the *PointStamped* message are augmented with the defined offsets for that robot. This sets the goal position of the follower robot, relative to the leader robot, in its defined formation position.
- This new *PointStamped* messge is then converted to a point in the map frame.
- The new *PointStamped* message is copied to a *PoseStamped* message to be used as the follower robot's goal pose.

Since a *PoseStamped* message has an additional orientation field, the orientation of the follower robot must be determined. The YOLO node's incoming *PointStamped* positions are used for this as they represent the position of the lead robot relative to the follower robot. To determine the orientation of the goal pose:

1. The yaw angle from the follower robot to the lead robot is determined from the formula:

$$\theta = tan^{-1}\frac{y_{target}}{x_{target}} \qquad (3)$$

2. The yaw angle determined is converted into a quaternion value by using the conversion function from *tf_transformations.quaternion_from_euler.*
3. This quaternion is added to the orientation field of the goal pose.

The goal pose is then passed to the follower robot's navigation node on the '/goal_pose' topic. The navigation stack will then use this goal pose to move the robot.

As the leader robot moves, the position of the lead robot moves in the follower robot's camera view. As it moves, the positions and distances are updated, setting waypoints of goal poses for the follower robot to follow.

5 Results and evaluation

The experimental trials were conducted in the Gazebo "City World" simulation, where the leader robot completed a full loop around an urban block. During each trial, the two follower robots detected and trailed the leader robot. The follower robots first trailed the leader robot in a line formation depicted in Figure 8a, and then in a delta formation depicted in Figure 8b, maintaining their prescribed offsets as the team navigated the course.

5.1 Algorithm performance metrics

The object detection performance in this project is evaluated using a set of standard metrics that capture both localisation accuracy and the robustness of the classification. The primary measures include Intersection over Union (IoU) to assess bounding box overlap and mean

Average Precision (mAP) to summarise the performance across all classes and thresholds. The Intersection over Union (IoU) measures the overlap between a predicted bounding box B_p and the corresponding ground-truth box B_{gt}:

$$\text{IoU} = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} \quad (4)$$

A prediction is a true positive (TP) if its IoU with a ground-truth box exceeds a specified threshold (commonly 0.50 or a value between 0.5 and 0.95).

5.1.1 Precision and recall

The precision indicates the proportion of predicted boxes that correctly match ground truth and is given by

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (5)$$

The recall indicates the proportion of ground-truth boxes that were successfully detected and is given by

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \quad (6)$$

5.1.2 Average precision (AP)

For each class, the predictions are ranked by confidence, and a precision-recall curve is generated by varying the confidence threshold. The Average Precision (AP) is computed as the area under this curve and is given by

$$\text{AP} = \int_0^1 P(r)\, dr, \quad (7)$$

where $P(r)$ is precision at recall value r.

5.1.3 Mean average precision (mAP)

The mean average precision (mAP) aggregates AP values across all classes (and sometimes at multiple IoU thresholds). Some examples are as follows:

- **mAP@50**
  AP averaged over classes at IoU = 0.50.
- **mAP@(50–95)**
  AP averaged over classes and IoU thresholds from 0.50 to 0.95 in 0.05 increments.

When evaluating object detection algorithms, it is common practice to report the performance by specifying the mAP@50 and mAP@(50-95).

5.2 Simulation test results

A representative example of a successful experimental trial is presented in Figure 9, which illustrates both the Gazebo simulation environment and the RViz visualisation.

In Figure 9a, all three robots are positioned at their initial starting locations, with the leader robot receiving a predefined sequence of waypoints for navigation. Figure 9b and Figure 9c depict the intermediate stages of the experiment, where the follower robots actively track and follow the leader as it progresses through the simulated environment.

Finally, Figure 9d captures the robots at the destination, with RViz confirming that the navigation goal has been successfully reached by all robots.

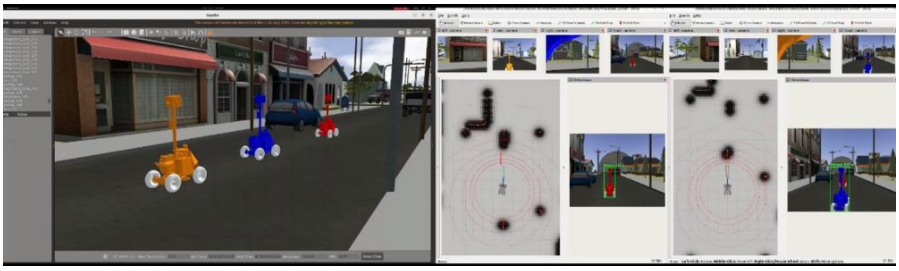
5.2.1 Object detection algorithm performance

Applying an off-the-shelf YOLOv8 checkpoint yielded a mAP@50 of 35.7 %, indicating that the network's generic feature representations do not generalise well to the quad-based dataset. The low scores across the available COCO (common objects in context) classes in Table 1 highlight this limitation. Since "robot" is not part of the pre-trained label set, the detector cannot directly recognise robots in the simulated scenarios.

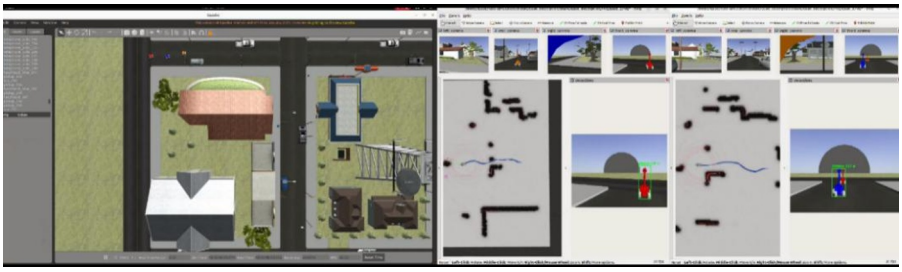
Table 1: Pre-trained model performance (mAP@50)

Class	Images	Instances	Box(P	R	mAP50	mAP50-95):
all	157	150	0.0452	0.232	0.0324	0.0284
person	29	29	0.0244	0.414	0.0339	0.0285
bicycle	29	29	0	0	0	0
car	22	22	0.00509	0.136	0.00424	0.00397
motorcycle	23	23	0.125	0.261	0.0795	0.0656
airplane	32	32	0.087	0.25	0.0564	0.0541
bus	15	15	0.0296	0.333	0.0203	0.0181

After fine-tuning on the annotated dataset, the custom YOLOv8 model achieves a mAP@50 of 92.3 % and an overall precision of 91.0 % at IoU = 0.5. Class detection results show red_robot_front at 94.2 %, red_robot_left at 92.7 %, and blue_robot_right at 91.4 %.

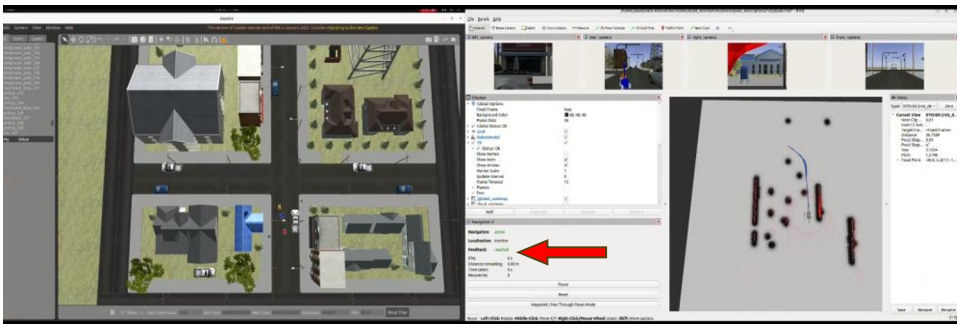


a) Robots in their starting positions in a linear formation



b) Robots travelling west-bound along planned route

c) Robots travelling east-bound towards end position along planned route



d) Robots at the destination. Rviz of the leader robot indicates that the final navigation goal has been reached.

Fig 9: Multi-robot navigation in a simulation world.

These high values demonstrate that training on domain-specific data is important for accurate detection in this application. The results for the model trained on the annotated robot image dataset is shown in Table 2.

Table 2: Custom-trained model performance (mAP@50 & Precision).

Class	Images	Instances	Box(P	R	mAP50	mAP50-95):
all	157	150	0.91	0.963	0.976	0.915
blue_robot_left	29	29	0.931	0.926	0.976	0.884
blue_robot_rear	29	29	0.935	0.99	0.987	0.952
blue_robot_right	22	22	0.745	0.864	0.914	0.848
red_robot_left	23	23	0.97	1	0.995	0.919
red_robot_rear	32	32	0.979	1	0.995	0.957
red_robot_right	15	15	0.899	1	0.991	0.933

The success of the navigation largely depends on the detections and distance determined from the YOLO node. For this reason, hardware used was a GPU type NVIDIA GeForce RTX 4070. This GPU allows the processing of the detections with a satisfactory speed for the follower robots.

6 Conclusion

This work employs a simulation of an autonomous, formation-controlled multi-robot system with minimal data exchange between robots. This formation control is achieved by using computer vision and LiDAR-based navigation in the follower robots, and a formation controller running in the lead robot that manages the formation of the robots as they traverse

through an environment. Additionally, each robot in the simulation has obstacle avoidance in its navigation stack, allowing them to navigate through an environment, keeping their formation, and avoiding potential collisions and dynamic obstacles along the way.

The work was conducted in the Gazebo simulation environment, using RViz, a 3D visualisation tool for ROS2. A model of the HERMES Security Quad robot used for the development of the work, which has a Velodyne VLP-16 multi-beam LiDAR, an MTi 630 Inertial Measurement Unit (IMU), an RTK GPS system and a multi-camera sensor box. This robot also has autonomous navigation and obstacle avoidance capabilities This robot in the simulation was triplicated and different colours are added to these clones to distinguish the three robots in the simulation. The main robot from which the robots were cloned from is the general leader, and the cloned robots are the follower robots.

Once these robots were in the simulation, a dataset of the different coloured robots was created from different visual angles to train a YOLOv8 computer vision model. This would allow one robot to recognise and detect a specified leader robot through its camera visuals. Once the computer vision model was trained, a robot recognition node was developed to be implemented into the simulated robots, which allowed the robot to determine the position of the leader robot relative to itself.

The general leader robot determines the formation in which the follower robots will travel. Once one of the follower robots detects the leader, it uses the leader's position information, derived from the recognition and detection node, to determine a goal pose. This goal pose is defined as an offset to the leader's position, which aligns with the position in the formation where the follower robot is supposed to be. The path and movements to this goal pose are governed by the navigation stack implemented on each robot.

Using this methodology, a multi-robot system has been implemented that can traverse an environment, keep a pre-determined formation, maintain minimal data exchange between robots, and avoid stationary and dynamic obstacles. The overall performance of the simulation could be improved by using more powerful computers, as running the multi-robot simulation on the current machines available led to very slow simulation times.

The robustness of YOLO detection in complex environments could be improved through dataset creation and management. Lastly, it would be necessary to extend the system to work on real-world mobile robot platforms in real-world scenarios to assess the system's actual

References

1. J. Shao, G. Xie, L. Wang, Leader-following formation control of multiple mobile vehicles. IET Control Theory Appl. **1**, 545–552 (2007). http://doi.org/10.1049/iet-cta:20050371
2. H. Wang, J.L. Wang, Leader-following formation control of multi-agent systems under fixed and switching topologies. Int. J. Control **85**, 695–705 (2012). https://doi.org/10.1080/00207179.2012.662720
3. K.T. Seow, M.A. Lewis, High precision formation control of mobile robots using virtual structures. Auton. Robots **4**, 387–403 (1997). https://doi.org/10.1023/A:1008814708459
4. C. Wu, A UHF RFID-based dynamic object following method for a mobile robot using phase difference information. IEEE Trans. Instrum. Meas. **70**, 1–11 (2021). https://doi.org/10.1109/TIM.2021.3073712
5. H. Zhang, J. Liu, Y. Wang, Y. Chen, R. Fan, Z. Miao, Vision-based formation control of mobile robots with FOV constraints and unknown feature depth. IEEE Trans. Control Syst. Technol. **29**, 2231–2238 (2021). https://doi.org/10.1109/TCST.2020.3023415

6. H. Poonawala, A. Satici, N. Gans, M. Wainwright, S. Hutchinson, Formation control of wheeled robots with vision-based position measurement, in Proceedings of the American Control Conference, Montreal, Canada (2012), pp. 4812–4817. https://doi.org/10.1109/ACC.2012.6315000
7. J. Hasan, D. Lim, Efficient robot tracking system using single-image-based object detection and position estimation. ICT Express **10**, 125–131 (2024). https://doi.org/10.1016/j.icte.2023.07.009
8. E. Montijano, D. Zhou, M. Schwager, C. Sagues, S. Martinez, Vision-based distributed formation control without an external positioning system. IEEE Trans. Robot. **32**, 339–351 (2016). https://doi.org/10.1109/TRO.2016.2523542
9. G. Zhang, C. Yu, L. Wang, D. Pu, Agile formation control of drone flocking enhanced with active vision-based relative localization. IEEE Robot. Autom. Lett. **7**, 6359–6366 (2022). https://doi.org/10.48550/arXiv.2108.05505
10. T. S. Dickens, Z. Smith, J. Tsegaye, Design of HERMES: a mobile autonomous platform, in Procedings of the RAPDASA-RobMech-PRASA-AMI Conference, Pretoria, South Africa (2023). https://doi.org/10.1051/matecconf/202338804005
11. Open Robotics, Simulate before you build. Open Robotics. [Online]. Available: https://gazebosim.org/home (Accessed 3 Apr. 2025).
12. Open Robotics, RViz User Guide. Open Robotics (2025). [Online]. Available: https://docs.ros.org/en/humble/Tutorials/Intermediate/RViz/RViz-User-Guide/RViz-User-Guide.html (Accessed 3 Apr. 2025).
13. Ultralytics Inc., Explore Ultralytics YOLOv8. Roboflow. [Online]. Available: https://docs.ultralytics.com (Accessed 28 Feb. 2025).
14. G.H. Hwang, S.W. Lee, J. Jeon, ROS2 implementation of object detection and distance estimation using camera and 2D LiDAR fusion in autonomous vehicle, in Proceedings of IEEE Int. Symp. Ind. Electron. (ISIE), Ulsan, Korea (2024)
15. Open Navigation, Groot – Interacting with Behavior Trees. Open Navigation (2023). [Online]. Available: https://docs.nav2.org/tutorials/docs/using_groot.html (Accessed 4 Apr. 2025).
16. S. Macenski, T. Moore, W. Roderick, G. Chitta, The Marathon 2: A navigation system, in Proceedings of the IEEE/RSJ Int. Conference, Intell. Robots Syst. (IROS), Las Vegas, USA (2020), pp. 3356–3363
17. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in Proceedings of IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, USA (2016), pp. 779–788
18. M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, XNOR-Net: ImageNet classification using binary convolutional neural networks in Proceedings of Eur. Conference, Computer Vision (ECCV), Amsterdam, Netherlands (2016), pp. 525–542