LIMO The Detector: A Service Robot for   
Object Detection

Ahmed Elbary  
Department of Computer Science  
University of LincolnLincoln, United Kingdom  
29385647@students.lincoln.ac.uk

*Abstract*—This paper introduces "LIMO Detector," a programmable robot that can recognise and count colourful things while navigating children's rooms on its own. The robot, based on ROS2 components and Python scripts for mapping, inspection, and navigation, performs well in a simulation. Colour object recognition and collision-free navigation are two important achievements. The system's effectiveness and scalability are confirmed by quantitative measurements, which also show that it has the potential for wider service robotics applications.

Keywords—Gazebo Simulation, Mobile Robots, OpenCV, Autonomous.

# Introduction

The idea of this project is to help parents detect and count their children's toys of different colours, the project developed a detection and counting of colour objects in two different simulated environments. Using the lidar provided by LIMO mobile robot helped to remap the rooms, and avoid obstacles when navigation was running.

In addition to two predefined worlds, the project presented good functionality in random rooms with some objects and obstacles whenever placed in the room. Some limitations in terms of the accuracy of object counting due to the simulation computational process and the noises might affect the counting process.

To justify the system, an evaluation has been done by comparing the number of detected objects many times in both worlds with the number of real objects number, which was a limitation in this project.

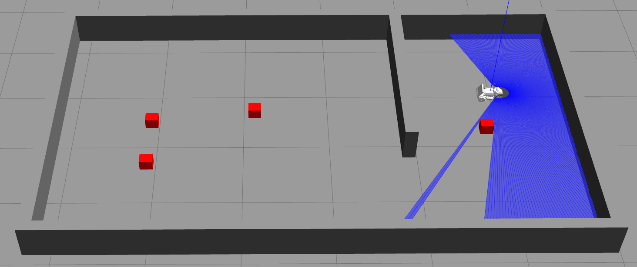
# Methodology

## Models identify

One cube object with a dimension of (0.15m × 0.15m × 0.15m) and two different colours (Red and Green) was created to be used as a detectable object. The use of solid colours (Red, Green, and Blue) has proven its simplicity in being detected by the camera sensor.

## Worlds construction

* Two different worlds were developed to test how well the system works regarding environmental complexity. In the basic world, four red boxes are placed in a pattern with only one wall as an obstacle.
* The more advanced world that has an additional wall such a way to hide some objects behind for navigation testing. In addition to the four red boxes, two more green boxes were added to test how the OpenCV library could help detect different colour ranges. Fig 1 shows the two worlds prepared in Gazebo Simulation software.



## 

Fig 1 a) The Basic World with four objects b)The Advanced World with additional objects and another wall

## Worlds inspection

### SLAM toolbox

Two different maps were created from the predefined worlds using the Simultaneous Localisation and Mapping toolbox (SLAM) to improve the accuracy of navigation, Fig 2 illustrates the result of the advanced map mapping as it presented the obstacles as a black entity while the discovered empty areas with white floor.

SLAM toolbox was the best approach regarding compatibility with ROS2 packages compared with other mapping approaches such as GMapping, and scalability compared to Cartographer [3].



Fig 2 the SLAM toolbox mapping for the advanced map



Fig 3 The inflation radius is shown in red colour.

### “ROS Node: SLAM toolbox is run in synchronous mode, which generates a ROS node. This node subscribes to laser scan and odometry topics and publishes a map to Odom transform and a map. Mapping: Laser scans associated with each pose in the pose graph are used to construct and publish a map.” [2]

### Navigation System

When designing the navigation system in this project, the challenge was to find a modular and ROS2 capability. In comparison to global mapping using a Dynamic Window Approach (DWA) and map-based navigation using predefined waypoints, the map-based approach was chosen due to the following factors:

#### Simplicity

The waypoint-based approach offers easiness for controlling the robot areas to cover, this approach is much more suitable for indoor environments such as children's rooms.

Firstly, the robot is set to be in an initial pose. Secondly, the rest points were found by moving the robot to different points with the 2D pose estimate tool provided by RViz. Finally, to get the output position and orientation of each point using the *amcl* topic provided with the ROS2 package.

#### Path planning efficiency

The use of the *BasicNavigator* class provided by the *nav2\_simple\_commander* library is ideal with the ROS2 navigation stack. This procedure allowed the robot to employ path planning algorithms such as A\* or RRT [4].

#### Parameters adjustability

In this method, the availability to modify the navigation parameters for more complex navigation tasks such as avoiding hitting obstacles by adjusting the inflation radius of the *local costmap*, as shown below in Fig 3, the inflation radius (in red colour) was increased from the default value of 0.1 to 0.25.

#### waypoints

As the map structure differs from one environment to another, two separate waypoint groups have been created for navigation tasks. Only six points were developed for the basic world inspection task; however, eleven points were for the advanced world.

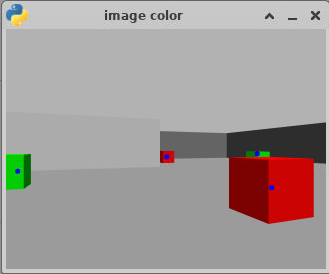


Fig 4 the functionality of detecting two different colours

The format for these points is as follows:

***[x\_pos, y\_pos, x\_ori, y\_ori, z\_ori, w\_ori]***

In which **(** ***x\_pos, y\_pos )* represent** the robot position in the  
X-axis and Y-axis accordingly, where **( *z\_pos* )** was ignored since the robot is only moving on the ground, while **(** ***x\_ori, y\_ori, z\_ori )* are**the quaternion representing the robot's orientation **around X, Y, and Z axes respectively, and finally the ( *w\_ori )* which is a scalar part** of the quaternion that fully defines the robot's orientation.

## Objects detection

For the robot to detect different coloured objects, two options were available to choose from: OpenCV library, and YOLO library, the former was chosen due to its low-cost computational process and its Colour Thresholding method of detection.

Two contrasting colour range values were used since two different coloured objects were (red and green) used in this project. For the red colour object, the colour range used was (0, 0, 80), (50, 50, 255) and for the green object (0, 80, 0),  
(50, 255, 50), as these values are for BGR format [1]. Fig 4 shows the functionality of the four objects detection processes.

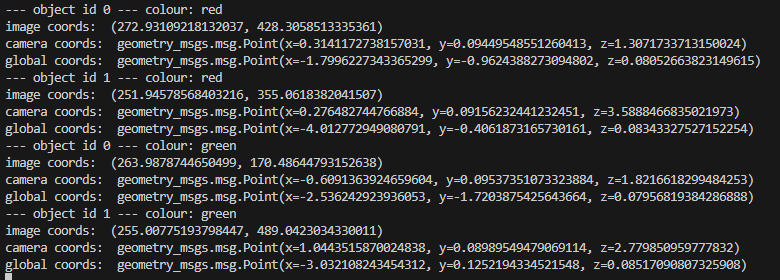
## Objects counter

One of the main issues in the object detection module was effectively counting objects that were near to each other, this difficulty emerged because the closeness of items sometimes prompted the detection system to interpret them as a single entity, resulting in inaccurate object counts.

To solve this problem, the detection threshold was set at 1.5 meters. This value sets the minimum distance at which objects are considered different from one to another. While this value may appear to be a big value at first, repeated testing revealed that it is the most effective.

# Results and Evaluation

The detector output shows three different coordinates, image coordinates, camera coordinates, and global coordinates of the real-time image view. The counter output shows the total of objects number during the whole navigation and the location on the map of each object. Fig 5 presents the detector and counter output respectively.



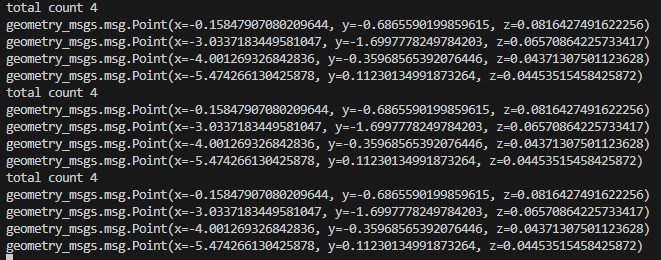


Figure 5 the four objects detecting and counting respectively

Future improvement is to create adaptive thresholds based on object sizes or add additional filtering methods to pick the exact object location while removing the others.

Several simulation trials were carried out, and it was found that the total number of objects detected changed with every trial. Object position, overlapping objects, and complex computational processes that happen during the detection process are some of the causes of this difference.

The outputs, which are collected in Table I, show that the system's object counting accuracy was acceptable. In particular, the detection accuracy was high in the basic map scenario, with 80% of properly detected objects.

However, the advanced map scenario produced a lower accuracy of 60%, mainly because the more complex environment needs more waypoints, which causes the robot to pause and adjust its position and direction more frequently, which may affect its field of view and object detection reliability; additionally, the physical complexity of the advanced map, such as the hidden area, further complicate these issues.

Table I dataset of detection outputs for both worlds

|  |  |  |
| --- | --- | --- |
| Trial | Basic world  (4 red objects) | Advance world  (6 red and green objects) |
| 1 | 4 objects detected | 10 objects detected |
| 2 | 4 objects detected | 6 objects detected |
| 3 | 4 objects detected | 6 objects detected |
| 4 | 6 objects detected | 5 objects detected |
| 5 | 4 objects detected | 6 objects detected |
| 6 | 4 objects detected | 6 objects detected |
| 7 | 4 objects detected | 6 objects detected |
| 8 | 3 objects detected | 6 objects detected |
| 9 | 4 objects detected | 8 objects detected |
| 10 | 4 objects detected | 6 objects detected |

Although the adjustable navigation parameters specifically the inflation radius, the path planning and the obstacle avoided weren’t as much as expected. One of the main reasons why more waypoints were used is that the robot navigation system chose a narrow path ignoring the wider space which led to obstacle hit several times.

Mapping with the SLAM toolbox is a reliable way to create the maps but, it might need more parameter adjustability for a chance of a higher precision map. Increasing the number of waypoints led to more stable robot navigation without hitting the walls and the objects.

# Conclusion

The "LIMO Detector" project effectively addresses the real-world difficulties parents have in their children's playrooms by showcasing the capabilities of a programmable robot for identifying and counting coloured toys. The robot was able to navigate, avoid obstacles, and map effectively in simulated environments by utilizing ROS2 components and Python programming. The system's excellent accuracy in simple scenarios was validated by quantitative evaluations, although computing restrictions and simulation noise caused slight performance limitations in more complicated environments.

The system's scalability was further demonstrated by the effective functioning in both organized and random layouts with LiDAR sensors and camera reading. The robot frequently produced reliable outputs, indicating its potential for wider service robotics applications, even if there were frequent errors in counting. To increase performance in the real world, future research might concentrate on improving detection accuracy and navigation processes.

Acknowledgement

*I extend my deepest gratitude to Prof. Grzegorz Cielniak and Dr. Riccardo Polvara for their guidance and support in carrying out this project. Sincerely thank the authors of the referenced papers for their inspiring research.*

##### References

1. OpenCV, “OpenCV: Changing Colorspaces,” *docs.opencv.org*. https://docs.opencv.org/4.x/df/d9d/tutorial\_py\_colorspaces.html (accessed Apr. 14, 2022).
2. S. Macenski, “GitHub - SteveMacenski/slam\_toolbox Parameters,” *GitHub*, Dec. 13, 2024. https://github.com/SteveMacenski/slam\_toolbox#toolbox-params (accessed Jan. 14, 2025).
3. S. D. Varanasi, Moulik Tammana, and Rajesh Kannan Megalingam, “Robotic Navigation Unveiled: A Comprehensive Study of GMapping, Hector Slam, and Cartographer,” *2024 3rd International Conference for Innovation in Technology (INOCON)*, pp. 1–6, Mar. 2024, doi: https://doi.org/10.1109/INOCON60754.2024.10511508.
4. Y. Li, “An RRT-Based Path Planning Strategy in a Dynamic Environment,” pp. 1–5, Feb. 2021, doi: https://doi.org/10.1109/icara51699.2021.9376472.
5. Bulat Abbyasov, R. Lavrenov, Aufar Zakiev, K. Yakovlev, Mikhail Svinin, and E. Magid, “Automatic tool for Gazebo world construction: from a grayscale image to a 3D solid model,” May 2020, doi: https://doi.org/10.1109/icra40945.2020.9