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

Robots are noticeably expected to work in dynamic environments and beside non-expert users. To meet this required demands, Robot Learning from Demonstration (LfD) has appeared as a promising approach that enables the robots to have new skills by observing the human behaviour. In contrast with the traditional programming methods that ask for manually controlling, the LfD offers a more flexible framework for learning more complex behaviours from the data given.

In the context of a trajectory learning, the LfD is often used to model time movement patterns that a robot can learn and repeat the tasks across varying scenarios [1]. Two main studied techniques in this area are: Dynamic Movement Primitives (DMPs) and Stable Estimator of Dynamical Systems (SEDS), where Both of them aim to produce demonstrated trajectories but also achieving stability and generalization, however they differ in their mathematical formulations and guarantees. In the DMP it represents the trajectories through dynamical systems that are modulated by forcing terms, which allow for a time scalability and. In contrast, the SEDS models a general stable dynamical system using Gaussian Mixture Models (GMMs) and applies convergence through the Lyapunov constraints.

This report aims to explore and compare both approaches using a set of 2D demonstrated trajectories of varying complexity between them. The goal is to discuss their theoretical foundations based on two relevant papers and their practical implementation, while the rest of the report applies both methods to a real dataset, highlighting key differences in stability, generalization, and performance.

# Literature Review

Robot Learning from Demonstration (LfD) enables the robots to learn more skills by understanding human provided instructions, avoiding the need for task specific programming for each scenario. This method has been widely used in robot manipulation, motion planning, and also in human-robot applications. Among the LfD methods two adopted frameworks have been widely used: Stable Estimator of Dynamical Systems (SEDS) and Dynamic Movement Primitives (DMPs) by demonstrating a strong ability for trajectory modelling where each of these methods has its own distinct advantages.

Khansari-Zadeh and Billard have introduced the SEDS framework in which described the motion learning as the estimation of a nonlinear time invariant dynamical system [1]. The system models the flow of velocity vectors across the state space using the Gaussian Mixture Model (GMM) and trained to produce the observed behaviour while also ensuring that all the trajectories is reaching to a desired goal. In contrast with the time dependent systems, this formulation enables robustness to perturbations and allows the system to respond dynamically without relying on time-indexed trajectories [1]. The key innovation of SEDS is its use of Lyapunov stability theory to embed global asymptotic stability constraints directly into the GMM parameter optimization process, which is framed as a convex optimization problem [1]. This formulation ensures theoretical convergence while remaining computationally efficient.

Demonstration data for SEDS typically consists of position and velocity pairs, obtained through kinesthetic teaching or teleoperation [1]. The output of learning is a stable, continuous velocity field guiding the system toward the goal. However, the model is sensitive to the number of GMM components, which must be tuned carefully to avoid overfitting or instability. Moreover, because SEDS is time-invariant, it lacks mechanisms for encoding time-dependent or phase-based behaviors, which can limit its adaptability in tasks requiring temporal modulation.

To situate SEDS within the broader landscape, Ravichandar et al. provide a comprehensive survey of LfD approaches, offering a taxonomy of methods based on their learning outputs, generalization capabilities, and demonstration modalities [2]. They distinguish between low-level motor primitives, like those used in SEDS and DMPs, and high-level task policies that abstract away from specific trajectories [2]. The survey emphasizes the strengths of SEDS in its interpretability, stability, and real-time applicability, particularly in industrial and collaborative robotics [2].

In contrast, DMPs represent motor skills using second-order differential equations with spring-damper dynamics, allowing for temporal modulation and smooth trajectory generation [2]. DMPs can adapt to different goal positions or durations of motion execution, making them suitable for tasks with changing temporal or spatial conditions. However, DMPs lack built-in stability guarantees, and extra effort is needed to ensure that trajectories do not diverge or exhibit oscillatory behavior under disturbance [2].

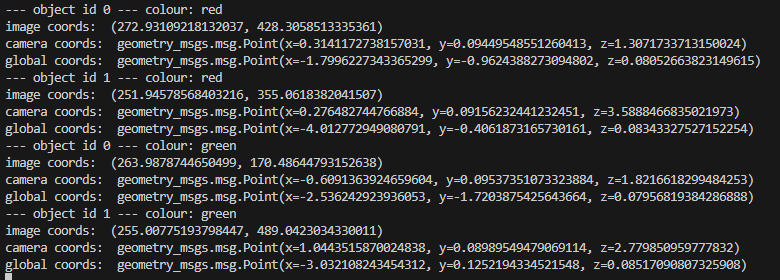
The survey also discusses the modalities through which demonstrations are introduced—passive observation, teleoperation, and kinesthetic teaching—and how these affect model performance [2]. For instance, kinesthetic teaching produces highly aligned input data, which suits SEDS’s stability-focused framework, but can be physically demanding and limited in scalability. DMPs, by contrast, offer more tolerance to noisy or less consistent demonstrations due to their dynamic modulation capabilities [2].

From a machine learning perspective, both SEDS and DMPs are classified as non-deep, data-efficient methods, prioritizing generalizability and transparency over raw model complexity [2]. This makes them especially appropriate for applications with limited training data or real-time control requirements, as in industrial settings or human-robot collaboration. In contrast, emerging deep imitation learning methods may offer better performance in unstructured or visual domains, but they sacrifice interpretability and demand greater computational resources.

In summary, SEDS provides a principled approach to learning converging motions from demonstrations, offering formal guarantees of stability, generalization across initial conditions, and efficient learning [1]. DMPs, while more flexible in encoding and adapting motion profiles, lack convergence assurances and require tuning to match task goals robustly [2]. These trade-offs form the basis for the comparative evaluation in Part 2 of this report, where both methods are applied to real-world 2D trajectory datasets and assessed in terms of accuracy, stability, and implementation effort.

# 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.



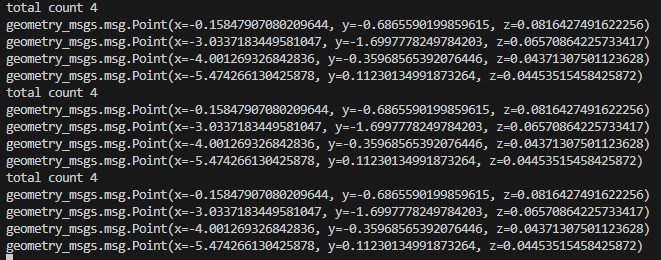


Figure 5 the four objects detecting and counting respectively

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. A. Noohian, M. Raisi and S. Khodaygan, "A Framework for Learning Dynamic Movement Primitives with Deep Reinforcement Learning," *2022 10th RSI International Conference on Robotics and Mechatronics (ICRoM)*, Tehran, Iran, Islamic Republic of, 2022, pp. 329-334, doi: 10.1109/ICRoM57054.2022.10025346.
2. S. M. Khansari-Zadeh and A. Billard, "Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models," in *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943-957, Oct. 2011, doi: 10.1109/TRO.2011.2159412.
3. H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, “Recent Advances in Robot Learning from Demonstration,” *Social Science Research Network*, vol. 3, no. 1, pp. 297–330, May 2020, doi: 10.1146/ANNUREV-CONTROL-100819-063206. Available: https://www.annualreviews.org/doi/pdf/10.1146/annurev-control-100819-063206