# Collaborative Multi-Robot Foraging using ROS and Gazebo

Module Code: COM00052H
Examination Number: Y388449

## I. Design

Autonomous robot foraging is a popular problem in the field of robotics [1]. Our problem involves retrieving various items scattered in a simulated environment. The items, red, green, and blue, have respective values of 5, 10, and 15. The further an item is from the robot homezone, the higher its value and the harder it is to find. Robots collect items by traversing through them and returning home. Item swapping, such as collecting a blue item while holding a red and returning home, brings a higher-value item closer to home and forms the basis of our solution.

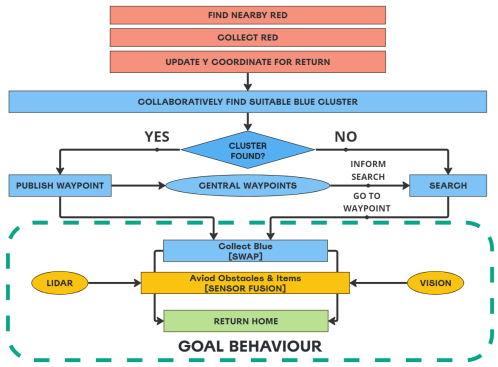


Fig. 1. Block diagram of system design to achieve goal behaviour

As per Fig1, The solution implements a collaborative homogenous multi-robot (2 robots) swapping system to reduce the time it takes to find blues by reaching a goal behaviour. Since red items spawn closest to home the system is designed to obtain a red item first. Robots then collaboratively find a blue item for swapping while avoiding greens. A multi-robot solution was elected to reduce the search time and allow better coverage than a single robot. However, even with multiple robots there is no guarantee that they will find a blue by themselves in a reasonable time and computation.

To reduce the time taken per robot to find blues of unknown location, inspiration was taken from local communication or leaving information in the environment [2] - specifically, ant foraging methods [3] that enable multi-robot collaboration via virtual pheromones or waypoints [4], [5], [6]. In this case, if any robot spots a suitable blue item - a virtual waypoint is shared centrally with other robots - robots then follow this trail to guide them towards a blue item, bypassing the random walk. Obtaining a blue item quickly is now guaranteed for all robots. The robots now swap the blue with the red and proceed to return home. Robots must avoid contact with items during this process. This is achieved by fusing lidar and vision data to simultaneously avoid items and obstacles, making up for the weakness of each other [7]. Once this cycle completes once, each robot should ideally have swapped the red in front of them with a blue. From now on robots only need to collect and return with this blue thus minimising the time it takes to find a blue while maximising the amount of blues collected.

## II. Implementation

The implementation of the aforementioned design was done using ROS2 Humble Hawksbill in Python plus related libraries and packages such as Nav2 for navigation. The solution uses a hybrid architecture with a custom finite state machine (FSM) [8] and Nav2 stack's behaviour trees (BT) [9], 10]. This ensured balance between modularity and reactivity. FSMs are generally known for their reactivity while BTs are better suited for modularity. Simpler high-level operations were abstracted into the FSM states for reactivity - ensuring we can reach a desired state from anywhere within the state machine. Complex deliberative operations such as navigating required the behaviour tree as they are generally considered better at handling recoveries from failure which is crucial when navigating [10].

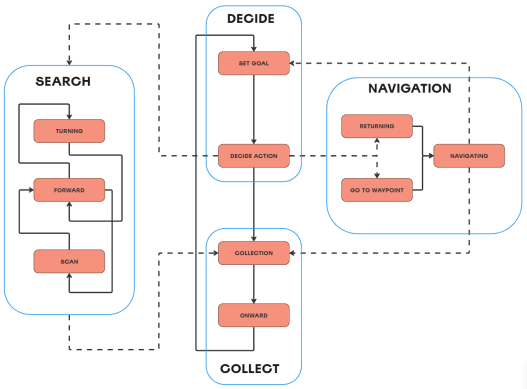


Fig. 2. Finite State Machine of implementation

Our system's architecture is based on a finite state machine (FSM) integrated within the robot_controller node, as illustrated in Figure 2. This design employs a set-decide-act paradigm, mirroring the traditional sense-plan-act framework [11], and divides its functionality into four primary super-states: Decide, Search, Collect, and Navigation. Each state is designed to handle specific tasks:

**Decide:** Determines the action state based on sensory inputs, selecting the target item and deciding between collection, searching, or returning home.

**Search:** Engages in item search while avoiding obstacles.

**Collect:** Directly collects items when identified.

**Navigation:** Manages waypoint navigation or homing, incorporating a recovery state for item drop scenarios.

Additionally, further robustness is introduced through the recovery state which has a transition from all action states.

The process begins with the SET_GOAL state, which selects a target item type based on the robot's current inventory, leveraging proprioceptive inputs. This selection is

facilitated through subscription to the /item_holder topic, tailored to each robot's identifier. Following this, the DECIDE_ACTION state evaluates prioritised actions (Navigate, Collect, Search) based on the identified target, transitioning to the appropriate action state as required.

The collection process involves approaching and retrieving the target, adjusting velocity based on item size and camera coordinates in the collect state and verifying collection upon reaching a predefined distance threshold in the onwards state. This draws inspiration from search and approach FSMs [12, pp. 55 – 61]]. If a suitable blue cluster with a clear path ahead is detected during collection, the robot's coordinates, yaw, and ID are published to a centralised itemmarks topic. A central topic for waypoints ensures better scalability and efficiency with multiple robots as it reduces overhead by eliminating the need for individual communications and enables filtering duplicate waypoints to the same cluster, mitigating confusion.

If no items are detected, the robot enters the search phase. It issues movement commands to explore and avoid obstacles, and can return to collection upon detecting a target. The SCAN state is specifically for target scanning. This state is implemented in addition to the FORWARD and TURNING states to improve the efficiency of the random walk search. If a waypoint update is detected while holding a red item, the robot transitions to the navigation phase. It subscribes to the /item_marks topic to navigate to the waypoint. After retrieving the item, the robot, if the item is blue, enters the RETURNING state and sets the PoseStamped to return home via Nav2.

The IsNavigating topic then activates the revised_scan node when navigating. Sensor complementarity is achieved as the node fuses camera and lidar data to simultaneously navigate and avoid obstacles, modifying laser scan outputs based on translated visual data. The IsNavigating topic enhances modularity and computational efficiency by only operating during navigation phases and by avoiding the need to frequently restart navigation systems upon item detection.

## III. Analysis

The performance metrics for foraging vary based on the problem at hand. Our solution aims to minimise the time taken to retrieve blue items by intelligently swapping items and reducing the search time through waypoints. Hence, the primary metrics were the time taken to retrieve subsequent blue items and the total number of items collected. The analysis was conducted using Python data science libraries and normalised when necessary.

To analyse system performance, we chose 6 random seeds based on their effectiveness in expanding situation coverage [[13]]. If necessary, we can add more seeds. This ensures a comprehensive yet efficient analysis protocol, achieved by combining Ahead of Run and In Run coverage [13]. The analysis took into account different configurations of item cluster placements. These configurations included close or distant red spawns, blue items either in line of sight or obscured by obstacles, item cluster density and path obstruction. Despite not testing all configurations, this diverse selection criterion guaranteed generalizability and reliability [13]. In addition to testing the system across environmental changes, it was compared to a benchmark. This was the same algorithm with two robots, but without waypoint collaboration. This was chosen over the naïve random walk to facilitate more accurate ablation studies. We anticipated that our solution would surpass the random walk, making any comparison lack nuance. All experiments ran for 10 minutes in real-time to track long-term trends and determine if the goal behaviour was achieved.

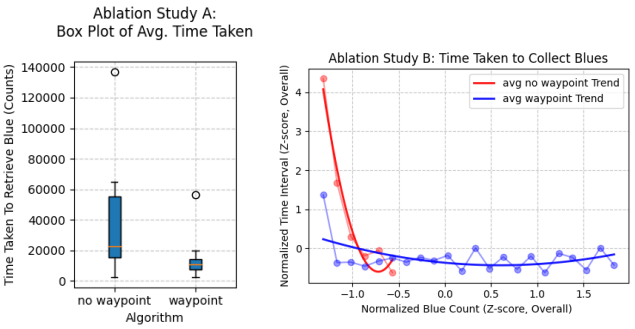


Fig. 3. Ablation Studies: Waypoints outperform benchmark

Fig 3 clearly shows that inclusion of the waypoint significantly reduces the time taken to collect blues in a more consistent manner as illustrated by the wider IQR range for 'no waypoint'. The outlier represents the time taken to collect the first blue. The inclusion of a waypoint successfully halves the time taken per robot. Normalising the time taken in Study B further supports this thus validating that waypoint publication catalyses finding blue per robot by bypassing the random walk.

Study B further shows disparity in total blue collected across the systems. The benchmark collects significantly less as robots spend more time searching than collecting across the same time frame. The z-score normalisation shows our solution performs significantly better with a consistently low time taken below the average in the long run after the relatively time consuming initial swap thus demonstrating the 'goal behaviour' our design intended.

Fig 4 demonstrates that this behaviour is consistent across seeds and that our solution works successfully.

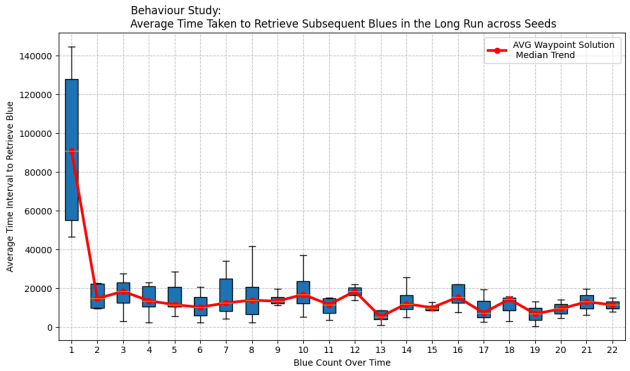


Fig. 4. The system successfully achieves the desired goal behaviour to swap and reduce time taken to collect blue in the long run

After an initially long time taken to perform swapping the subsequent times to collect blue is consistently minimised. The solution consistently engages in a loop-like behaviour to collect a nearby blue item after one swap, successfully avoiding other items. However, certain trends are particularly noteworthy. Firstly, the interquartile range (IQR) for the time taken to retrieve the first blue item varies significantly. Secondly, even though the time taken remains

consistently low afterward, there are fluctuations. These variations suggest that certain cluster configurations may be more compatible with the algorithm, but could also indicate potential flaws in the system.

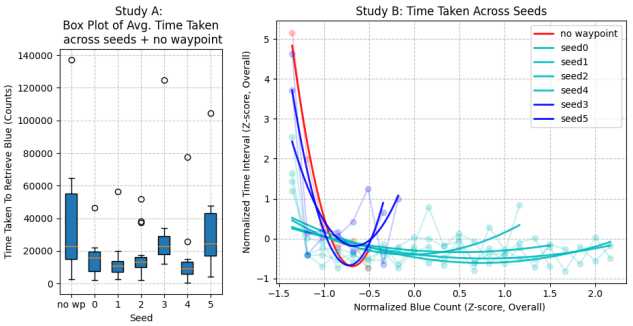


Fig. 5. Deeper dive into inconsistencies across seeds

As per Fig 5, Seeds 3 and 5 underperformed, with their times to locate the first blue nearly matching the solution without waypoints. This indicates that in certain configurations, waypoints have minimal effect. If one robot cannot locate the item quickly, none of them can. Both Seeds 3 and 5 had blue and green clusters hidden behind obstacles, which explains the first trend in Fig 4. Cluster placements hindering shortest paths and forcing robots to take longer routes after collecting the first blue item also negatively affected the trend. In Study B, these seeds also collected fewer blue items. A consistent explanation is that due to the proximity of red clusters, robots obstructed each other more than collecting items, causing minor collisions as depicted in Fig 6.

Additionally, the simulation incorporates a level of randomness, occasionally respawning blues at a different point or behind a red, which explains the second trend in Fig 4. Unexpectedly, robots deviated from the plan and collected a red and swapped an additional blue. This meant now there are 2 or more blues nearer to home and counterintuitively helped mitigate the randomness instead. Seeds 2 and 4 show this as there are multiple outliers representing swaps in Study A. These outliers should result in a wider IQR but conversely they helped narrow the subsequent time taken by alleviating randomness. This can also be mapped to Fig 4 where after deviations at counts 8 and 10 the immediate next collections were more consistent (narrower IQR). Hence, the solution might be more efficient with multiple initial blue swaps, potentially increasing blue collection consistency over time.

## IV. Evaluation

| Factor | Metric | Test | Result |
|---|---|---|---|
| Collision O -R - I | % collision counts | % Collision counts between robots, obstacles and items across x10 seeds | 0% obstacle collisions<br>9% robot collisions<br>18% item collision |
| Collect & Return | % success rate | Ran system to collect and return items across x10 seeds | 96% collection rate<br>89% return rate |
| Scalability | Average real time factor | Ran system with 1, 2 & 3 robots in gazebo x10 | 0.85 at 1 robot<br>0.67 at 2 robot<br>0.33 at 3 robot |
| Swap Recovery | % recovery rate | % recovery counts after item collision across x10 seeds | 98% recovery rate |

Fig. 6. Testing Summary across other metrics

In addition to the analysis of the system in relation to its intended design, alternative metrics to evaluate the system as a whole for the task of item retrieval were also taken into account such as collision avoidance, reliability of item collection, scalability, and swap recovery were metricated to provide a comprehensive performance evaluation to better understand the strengths and weaknesses in the real world. Understandably, some aspects of the solution such as items swapping close to home will not be translated into reality but allows for some interesting analysis as shown above.

Overall, the implementation performed the designed system as intended as seen in the analysis section. The implementation of central waypoints successfully catalysed the collection of total blues by minimising the time taken to find a high value item via multi-robots in a semi-unknown environment. The design of our solution consistently collects red and finds a high value item of unknown location in significantly reduced time. This aspect of collaboration to collect a red item and bring it to a blue target quickly while avoiding obstacles has implications in real world fields where time is critical but location is unknown such as search and rescue operations where care packages are distributed to victims of disaster while avoiding debris or even fields in agriculture where chemicals need to be collected and sprayed at target locations.The system also dynamically adapts the target and decisions based on how the space around it changes. Hence, it demonstrates limited ability to adapt to a dynamic and changing environment, for example if someone moves an item to a different location it autonomously decides to scan for it as it expects it.

Additionally, the self organised design allows us to achieve a sophisticated task such as collaborative item swapping emergently and with less computation by completing simpler sub actions akin to swarm robotics . However, swarm-based and decentralised paradigms are more complicated to deploy in practice. Due to uncertainty, emergent behaviours are rarely defined top down but require exhaustive iteration to reduce the reality gap [14] through automatic controller design via evolutionary algorithms or swarm languages like Buzz. Without this, gaps in our design and implementation could occur. This was evident in our analysis and the 89% return rate in Fig 6. During certain runs across seed 3, the robot could not return with a red item despite reaching the waypoint, because all the blues had already been swapped. The state machine logic deadlocked due to the lack of a transition to account for the combinatorial nature of the design space. This issue has broader implications for real-world scenarios, such as when environmental hazards are present at a waypoint but the robot heading towards it is unaware and ends up damaged.

The implementation of waypoints in practice would require communication channels such as wifi networks which have range limitations unlike our central waypoint topic implementation in ROS. Ideally, a central or decentral logic would need to be introduced to ensure robots always maintain a certain threshold distance between them so that communication is not disrupted [4, 5]. The topics could also be better implemented using ROS services and clients. Multi-robot collaboration is also limited as robots do not deliberate not going for the same item via auctions or voting which resulted in robot collisions when going for the same cluster. However, the solution was designed under the

assumption that red clusters would not spawn close to each other.

Additionally, complimentary sensor fusion between LIDAR and camera data allows for seamless navigation across items and obstacles. One significant limitation of the problem that widens the reality gap [14] was the inability to detect items via LIDAR and obstacles via camera. The solution addresses this effectively - multiple instances where the robot successfully weaves through clusters of items. However, the translation of vision data to LIDAR scan is not perfect. The estimation of the distance and angle from camera frames are precise to around 85% - while it takes exact focal length and actual item size into account it is unable to account for the optical flow, delays in image data callback, sensor noise and the lack of depth perception via 3D cameras. This resulted in cases where the robot did bump into another item and dropped the blue as shown in Fig 6. Thankfully, to recover from this the implementation includes a state to swap the item back. In simulation, this means simply waiting till the items automatically swap back in 5 seconds. However, in reality the robot would require an end effector to grasp objects and take that into account when avoiding obstacles on the way back. Our implementation does not account for this or the complicated physics involved since the items do not roll upon contact. In terms of the hardware, LIDAR is unable to detect obstacles such as glass and the monocular camera does not efficiently detect contours and depth or even multiple planes.

By using a known map of the environment our solution can reliably implement grid based navigation that is simple to implement. While this works for our solution in reality it is not adaptable as the dimension and space would increase across different environments and our grid based approach becomes computationally expensive. We would also need to create a map for each new location. To solve this, SLAM based algorithms are typically used [15]. Despite the imprecise estimates, in relation to the task of collecting and retrieving items the implementation performs them consistently with over 90% success rate in Fig 6. However, on rare occasions (4%), the robots will end up not collecting a target item when approaching it. This is due to the state machine prioritising obstacle avoidance over item collection when the items spawn too close to the obstacles.

Lastly there are concerns with how the real time factor affects the performance of our solution in the real world. Our solution ran at an average RTF of 0.67. This means the system would be slower to react to the environment and sensor readings. However, this is still an acceptable level as we can define thresholds to account for this delay. Since our problem is mostly in a static environment this RFT is sufficient.

## V. Safety and Ethics

Autonomous robotic systems need to be deployed in a safe and ethical manner. This includes being aware of ethical concerns, practices and guidelines when designing systems to alleviate ethical concerns as well as mitigate any risks to safety. Safety in robotic autonomous systems is achieved through rigour in situation based testing [13] and ensuring systems adhere to ethical principles [17]. This means asking ourselves if we searched well enough when testing and analysing our solution. Essentially how do we know when to stop testing and ensure our solution is safe and reliable. This is an open area of research as autonomy in safety and ethics are difficult to achieve. For example, when designing pick and place industrial robots the primary concern is efficiency and performing the task as fast and reliably as possible. Typically advanced machine learning approaches are used to generate controllers and object detection for these complex pick and place procedures. This practice is a concern as industrial robots typically work in factory settings with other workers. And therefore this needs to be taken into consideration when training the models. Without such considerations, there have been multiple reports of incidents where robotic arms move quickly and unexpectedly around humans and injure them [16].

Similarly, there were instances where image classification algorithms mistakenly detected people as objects to collect posing further risks to workers . This problem is particularly sensitive as vision algorithms suffer from feature bias in training. If AVs have been trained on an ethnically biassed set of images they may be more likely to fail to recognise certain ethnic groups as human (due to skin colour or clothing, for example), and make decisions that place such groups at greater risk [18]. These issues persist due to a lack of rigour as well as a level of unpredictability in situation based testing around humans who act unpredictably. These robots are simply victims of testing that lacked rigour. In other cases, it also alludes to a lack of realistic simulation environments that depict the randomness of human behaviour.[17] In contrast we also encounter ethical and safety concerns when robots perform a task too efficiently - which can have negative implications on workers around them. In the context of human-robot interaction, particularly cobots we need to ensure robots work at a pace that complements human workers. There have been multiple reports of workers unable to keep up with the pace of their robotic counterparts which results in long term physical and mental health risks.

Recently, there has been a surge of autonomous item retrieval fleets within the context of warehouse automation. Amazon, Kinova and Tesla are popular examples where multiple robots rely on AI based centralised fleet management systems to collect and sort goods [18]. This raises ethical concerns not only in terms of unemployment due to the displacement of workers usually among the working class but further delves into the concerns of opacity and oversight. What governs these systems to make decisions and prioritise the delivery of goods. In the immediate future given how ubiquitous robotic fleets are projected to be we need to be asking who will benefit because of these large fleets working in unison to effectively drive the economy. The decision making behind these processes are only becoming more complex and less transparent. Hence we need to ask ourselves what we are designing for - safety or efficiency. The answer should ideally be both.

## References


1] A. F. T. Winfield, ‘Foraging Robots’, in Encyclopedia of Complexity and Systems Science, R. A. Meyers, Ed. New York, NY: Springer New York, 2009, pp. 3682–3700[Online]. Availablehttps://doi.org/10.1007/978-0-387-30440-3_217.

[2] R. C. Arkin, Behavior-based Robotics. MIT Press, 1998[Online]. Availablehttps://play.google.com/store/books/details?id=mRWT6alZt9oC.

[3] B. Hölldobler and E. O. Wilson, ‘The Ants Springer-Verlag’. Berlin, 1990.

[4] R. T. Vaughan et al., ‘Blazing a trail: Insect-inspired resource transportation by a robot team’, in Distributed Autonomous Robotic Systems 4, L. E. Parker, G. Bekey, and J. Barhen, Eds. Tokyo: Springer Japan, 2000, pp. 111–120[Online]. Availablehttps://doi.org/10.1007/978-4-431-67919-6_11.

[5] R. T. Vaughan et al., ‘Whistling in the dark: cooperative trail following in uncertain localization space’, in Proceedings of the fourth international conference on Autonomous agents, Barcelona, Spain, 2000, pp. 187–194[Online]. Available Https://doi.org/10.1145/336595.337351[Accessed: 8 February 2024].

[6] ‘Multi-robot foraging for swarms of simple robots’, 2011[Online]. Available Https://search.proquest.com/openview/b3e05c7b108f9e7d3163b4b7edc63e01/1?pq-orgsite=scholar cbl=18750.

[7] S. G. Tzafestas, ‘12 - Mobile Robot Localization and Mapping’, in Introduction to Mobile Robot Control, S. G. Tzafestas, Ed. Oxford: Elsevier, 2014, pp. 479–531[Online]. Available Https://www.sciencedirect.com/science/article/pii/B9780124170490000122.

[8] M. Ben-Ari, Principles of Concurrent and Distributed Programming, Second Edition. Addison-Wesley Professional, 2006[Online]. Availablehttps://play.google.com/store/books/details?id=yXSKzwEACAAJ.

[9] M. Colledanchise and P. Ögren, ‘Behavior Trees in Robotics and AI: An Introduction’, arXiv [cs.RO], Aug. 31, 2017[Online]. Available Http://arxiv.org/abs/1709.00084.

[10] S. Castro, (May. 08, 2021), ‘Introduction to behaviour trees’, Robotic Sea Bass - Learn assorted topics in robotics, AI, programming, and more. [Online]. Available Https://roboticseabass.com/2021/05/08/introduction-to-behavior-trees/ [Accessed: 9 February 2024].

[11] A. Srivastava, ‘Sense-plan-act in robotic applications’. Unpublished, 2019[Online]. Available Http://rgdoi.net/10.13140/RG.2.2.21308.36481.

[12] M. Ben-Ari and F. Mondada, Elements of Robotics. Springer, 2017[Online]. Availablehttps://play.google.com/store/books/details?id=itpCDwAAQBAJ.

[13] R. Alexander et al., ‘Situation coverage – a coverage criterion for testing autonomous robots’, vol. Report number YCS-20, p. 21, Feb. 2015[Online]. Availablehttps://eprints.whiterose.ac.uk/88736/1/YCS_2015_496.pdf[Accessed: 9 February 2024].

[14] N. Jakobi et al., ‘Noise and the reality gap: The use of simulation in evolutionary robotics’, in Advances in Artificial Life, 1995, pp. 704–720[Online]. Availablehttp://dx.doi.org/10.1007/3-540-59496-5_337.

[15] H. Durrant-Whyte and T. Bailey, ‘Simultaneous localization and mapping: part I’, IEEE Robot. Autom. Mag., vol. 13, no. 2, pp. 99–110, Jun. 2006[Online]. Availablehttp://dx.doi.org/10.1109/MRA.2006.1638022.

[16] (2023, November.9), Industrial robot crushes worker to death as he checks whether it was working properly, CBS News. [Online]. Available: https://www.cbsnews.com/news/industrial-robot-crushes-worker-dead-south-korea/. [Accessed: 10 Feb. 2024].

[17] Ethical Issues for Robotics and Autonomous Systems. [Online]. Available: https://www.ukras.org.uk/wp-content/uploads/2021/01/UKRASWP_EthicalIssues2019_online.

[18] Y. Shen et al., ‘Multi-agent reinforcement learning for resource allocation in large-scale robotic warehouse sortation centers’, 2023[Online]. Availablehttps://assets.amazon.science/4b/78/fde3f0284c5084977c8173392270/multi-agent-reinforcement-learning-for-resource-allocation-in-large-scale-robotic-warehouse-sortation-centers.pdf[Accessed: 10February2024].