COMP 4034 Assignment Written Report

Student ID : 14306556
Group number : 10

***Abstract*—In this paper, we propose a robust solution to the object search problem. In the field of autonomous robotics, the object search problem is defined as an exploration task to locate and beacon towards a number of pre-defined objects. Our approach focuses on a generalised and modular strategy to maximise performance and usability in unknown environments with the possibility of extension to an unbounded number of classified objects. The implemented behavior utilities on-board Lidar and Kinect sensors in order to traverse a mapped environment and identify a series of goals. Methods include the use of frontier exploration, CNN object detection (YOLO) and point cloud pose estimation in conjunction with a navigation stack. In-depth system evaluations have been conducted in a simulated Gazebo environment with appropriate metrics to demonstrate relative performance and real-world feasibility.**

I. Introduction

For the average person, the object search problem is considered a simple task and is often solved with ease in daily life. For instance, finding a pen and paper. Our subconscious memory allows us to store this information, retrieve it when necessary and simply move towards the intended goal.

In the context of robotics, significant effort is required to manage related data streams to locate, classify and beacon towards a desired object. Challenges arise with both obstacles and unreachable locations and a controller should deal with these impediments accordingly. The scope of this project covers a level of autonomy above teleoperation and follows a set of pre-scripted behaviours, however once the program has started, the robot acts as a fully autonomous system.

This search by navigation problem focuses on robot movement in order to gain visibility [1]. Considering a mobile approach allows an increase in data collection from the environment to aid in searching and is essential to locating hidden objects (i.e. located in another room).

With respect to the problem a system should be able to perform the following actions to be considered functional:

A. Explore and gain coverage of the environment

B. Detect the given objects (we will use a green box, fire hydrant and mailbox)

C. Beacon towards ‘new’ objects only

D. Avoid obstacles whilst traversing the environment or beaconing

In our proposed method, we have abstracted these functional requirements into appropriate states. The controller then alters the current state via a state manager to perform actions according to sensor input.

There are several key metrics that should be considered and these will be discussed later when evaluating the performance.

Details of the constraints that were used throughout the implementation and testing are as follows:

- Software - ROS 1 (and associated packages), gazebo, rvis, python 2.7
- Robot - turtlebot waffle (differential drive and rectangular footprint)
- Sensors - lds_lfcd sensor (LIDAR), realsense_R200 (RGB/Depth)
- Prior information - map file produced by the gmapping ROS package

II. Methodology

During runtime, the robot remains in one of four distinct states: Frontier, Beacon, Obstacle Avoidance and Random Walk. Control flow is demonstrated in figure 1 and is managed by a State Manager class.

Initially, the robot will enter the Frontier state. This is considered the primary exploration state and the robot will move to boundary locations (frontiers) while the active goal list is empty, until there are no more available frontiers.

If an entry in the active goal list exists, the robot will enter the Beacon state. The robot will attempt to beacon towards a list of specified goals until no more active goal objectives are listed.

In the case that there are no available frontiers, the robot will alternate between secondary exploration states: Obstacle Avoidance and Random Walk. This condition ensures that more effective ways of exploration are prioritized but a continuation method remains in place to obtain any previously missed objects.

Information is printed to the console to provide insight to the user (e.g. current state/action).

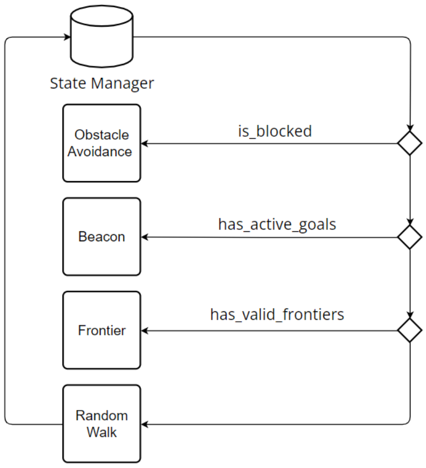

Figure 1. State Selection Flow Diagram

By recalling the original functional requirements, we can exploit the modularity in our solution to understand how the conditions for these state changes are determined and how individual state functionalities are achieved.

A. *Exploration (Frontier Algorithm)*

In our approach, an implementation of frontier was developed as the robot’s primary method of map exploration. Frontiers are defined as a set of cells that mark a boundary between ‘open’ and unknown cells [2]. This boundary allows ranges of unoccupied cells between obstacles that should be explored.

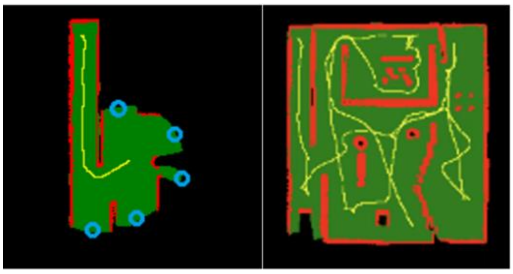

Figure 2. Example of frontier locations (left), Completed occupancy grid (right)

The algorithm uses an array representing an occupancy grid and two separate arrays to keep valid and invalid frontiers in memory. The occupancy grid maintains the probabilities of occupied cells and these are updated by the published laser scan. Frontiers are marked as invalid if move base returns an invalid state and will not be revisited.

Frontiers are prioritised by considering the Manhattan distance, a weighting (based on the number of occupied grid cells and lethal costmap cells between the robot and frontier) and a ratio of surrounding cells (open vs unknown). The aim of this prioritisation is to evaluate closer frontiers and frontiers that will provide the most value to the system first, resulting in efficient exploration.

After calculating the optimal frontier, the aim is to set a waypoint to that location in order to explore the selected and surrounding cells. As a result of a high-resolution occupancy grid, it is often seen that frontier cells are grouped close together. A clustering technique was used so that move base can consider frontier cells as one distinct location to prevent attempts to move to each individual cell.

It should be mentioned that this manipulation of a large grid becomes computationally expensive and a vectorised solution was implemented in order to run efficiently.

Figure 2 shows an example visualisation of an occupancy grid. Cell values are bounded by thresholds in the visualisation to represent cell types. The visualisation provides insight into how the robot has traversed the environment and the decisions made as it updates in real-time.

The objective of this component is to output a world co-ordinate that can be provided to the navigation stack to effectively traverse the map.

In the instance that there are no more valid frontiers (i.e. every cell is explored or unreachable) the robot's exploration behaviour defaults to a combination of Random Walk and Obstacle Avoidance. These states are low in complexity and aim to continue to traverse the map in a controlled manner.

Random Walk moves the robot forward a specified distance and then rotates to a random direction. *Obstacle Avoidance* inhibits this behaviour if an obstacle is detected within a threshold distance and will stop and rotate until no obstacles are within this distance.

The pair of states are inefficient due to a non-systematic and random approach to exploration, however over a long period of time the environment will be fully explored.

B. Object Detection (CNN)

Two main approaches were considered to solve the object detection problem, colour slicing, and a CNN based detection.

For the given task, distinguishing between objects (i.e. the green box, red fire hydrant and blue mailbox) is a relatively easily task by thresholding hue values. The method quickly shows flaws when considering different goal objects, features of the environment or obstacles of similar hue. While it may be possible to develop thresholds, present noise that may be resistant to image processing techniques can cause difficulties.

As an alternative approach, we chose to use a convolutional neural network (CNN) to detect and classify the objects present in sight. An established open-source framework exists as a ROS package (darknet_ros) to aid in training such models [3].

At a glance, an algorithm (YOLO) divides an image into a number of grid cells, each cell being responsible for its own information. The algorithm then uses a combination of these grid cells and a probability threshold to predict an object's existence [4]. This allows the model to return a bounding box and an associated classification.

In order for the model to recognise the features of an object, model weights must be trained by supplying data and performing a forward phase, followed by backpropagation. In our case, as we are using a simulated environment, data for our objects does not exist in online databases and therefore a data collection step was required.

A script was developed to save images to disk, that would later be labelled to build our custom database. Quality of these images were important for a high-performance model. A proof-of-concept model was trained with a database of 300 images, that was later expanded to over 1500 images. Weights were saved every 1000 iterations.

The images were labelled such that the bounding box should represent the main feature of the object. For example, the mailbox is identified by only the body so that gaps within the bounding box are not widely present. This decision was made to simplify object pose estimation.

The database was partitioned (90/10) and cross validation was performed in order to determine the best set of weights. Models were trained with a number of different hyper parameters (batch sizes, iterations etc.) and tuned using grid search. Various data augmentation techniques were also used (random cut-out, hue, rescaling) in order to improve the robustness of the model.

The objective of this component is to correctly locate and classify objects in sight and return a set of bounding boxes to be used by the pose estimation component.

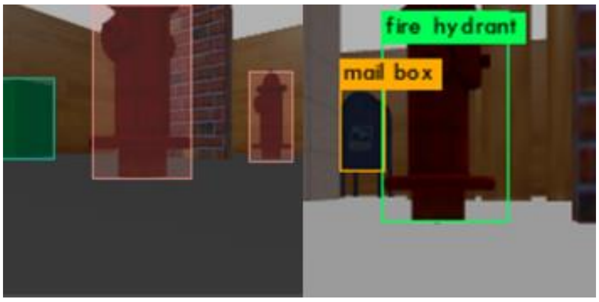

Figure 3. Example of self labelled objects (left), Example of model prediction (right)

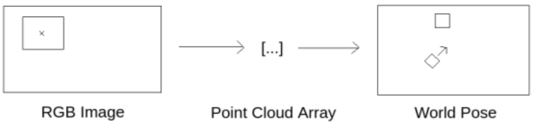

Figure 4. Representation of pose estimation approach

C. Determining 'new' objects (Pose Estimation)

Data fusion of the RGB and Depth image allows us to calculate the relative x and z distances of the detected object.

Given a bounding box, we can pinpoint the surface of an object in an RGB image by calculating the centre pixel. Data fusion is then performed by matching the resolution of the images, subscribing to both data streams and indexing into the 1D point cloud array using the x and y centre pixel.

Using the values returned, we can form a triangle to calculate the relative angle at which the object is detected. In combination with the robot's yaw, a resultant angle in the world can be drawn.

This resultant angle can be abstracted into world quadrants in order to retrieve the coefficients used in successive trigonometry calculations to calculate relative world co-ordinates. The summation of these co-ordinates with the robot's pose gives a pose estimate for the detected object. This methodology works for multiple objects in sight and all poses can be estimated by looping through all bounding boxes.

The overall solution becomes the fusion of robot odometry and sensor information. To aid in the data fusion and reduce the errors between different frequency callbacks, the *ApproximateTimeSynchroniser* policy is used here to combine messages into a single callback.

To determine whether a detected object is a 'new' object, we can compare the centre pose as this allows detection from any angle. It should be noted that a margin for error is accounted for by an error threshold.

The centre pose can be calculated by adding a radius offset (based on classified object) and an objective position can be calculated by subtracting a sufficient distance.

New objects are stored as goal objects in a goal manager, holding an associated class, centre and objective pose co-ordinates. The purpose of the goal manager is to save a list of distinct object locations and a world co-ordinate that the robot should beacon to in order to consider the goal as completed.

The objective of this component is to output a world co-ordinate that can be provided to the navigation stack to beacon towards a goal.

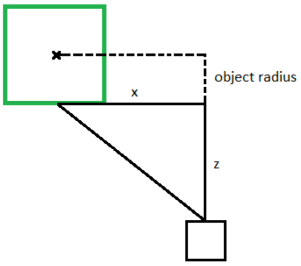

Figure 5. Representation of centre pose estimation

D. Path Planning (Navigation Stack)

A method of navigation was required to travel to a desired location (i.e. selected frontier or goal objective co-ordinates) As an alternative to a reactive solution (e.g. PID controller), a navigation stack was integrated to allow a robust pathing using move base. The main idea is that using a provided map, the robot creates an initial global costmap and a local costmap is created as a rolling window around the robot's location. Using AMCL, a probabilistic localisation system [5], the pose of the robot is estimated as it traverses the known map. The pair of costmaps are then used by local and global planners to path and navigate a valid course.

The global planner uses the global costmap to set an initial path, taking into account known 'lethal' cells. The robot attempts to follow this path to the objective.

The local planner uses the local costmap, which is updated by the robot's sensor information, and the provided global path to determine velocity commands to control the robot's trajectory.

The choice of local and global planners was an important step as these determine the methods of pathing. Two local planners were compared and evaluated: dwa_local_planner and teb_local_planner. The sample-based approach of dwa was seen to often cause a stagnation in local minima and would struggle with in-place rotations. The time-based approach of teb and the ability to alter the global path was shown to be robust and efficient. After evaluation, the default global planner (navfn) proved sufficient in conjunction with the teb_local planner.

Costmap parameters were tuned (inflation layer and cost factor) [6] to aid navigation through corridors and narrow spaces. Local planner parameters were based on results of multiple evaluations.

The objective of this component is to efficiently navigate to a world co-ordinate that is provided by frontier and the goal manager.

III. Evaluation

Influence of system parameters

During implementation, it was essential to evaluate the influence of system parameters in order to select optimal values. It should be noted that there are often associated compromises when tuning a parameter.

Linear velocity impacts the time taken for the robot to travel from point A to point B. To measure the effects of velocity, we recorded the number of cells discovered against time taken as the primary metric. Figure 6 shows that low velocities were slow to explore the map whilst high velocities also had low coverage due to uncontrollable actions and errors due to move base. Optimal values for the linear velocity parameters were proved to be between 0.2 and 0.3.

Angular velocity impacts point cloud distortion. As seen in figure 7 the error is propagated as a result of an increased angular velocity. In order to turn and navigate efficiently, a compromise must be made and a value was selected in the mid-range (1.0 radians per second).

Model weights impact the classification accuracy of the convolutional neural network. Figure 8 shows a confusion matrix for the green box classification with a chosen weight setting. Precision, recall and accuracy can be used to compare the classification performance. In this case, we want to maximise precision while maintaining a reasonable recall. This is to minimise false positives and reduce beaconing to incorrect objects. We used a F-beta measure (using the average precision and recall over the three objects) with a beta value of 0.5 to return the optimal set of weights (6000 iterations).

Performance evaluation

As shown previously, components were evaluated individually, however integration testing of the system was important to understand the overall performance and cohesion between features.

Test maps were created with varying complexity and intention to evaluate generality and test situational performance. There were key metrics that were used to evaluate the performance of our system between 5-minute runs in the test maps: percentage of correctly classified objects of all types reached, percentage of correctly classified objects of each type reached, number of incorrect classified objects reached. Any test runs that resulted in a collision were flagged as this event should not be tolerated on any occasion. This methodology allowed us to pinpoint limitations and areas of improvement.

In the benchmark test given in the demonstration, the system successfully explored the majority of the map in the given time, although it should be noted final room was not yet reached. All objects that were located in the map coverage were detected and all but one were beaconed. Due to the can layout, move base set a waypoint on a lethal costmap cell and was later cancelled due to inaccessibility. This would usually be corrected, however due to the lack of sparsity, the system failed to find another valid point. Throughout, the system avoided all obstacles.

By comparing against the original objectives of the system we can say that the approach completes all functional tasks to a high level.

Assessment of system limitations

Limitations include system implementation, architecture design and hardware constraints. We attempted to couple each limitation with an alternative solution to enhance performance. In this section, we will focus primarily on the most important limitations.

i) Frontier – Manhattan distance was used to calculate the optimal frontier

With the selected resolution, graph traversal algorithms such as a* could not be included as several seconds were required to compute the next frontier. This forced the use a of a non-optimal solution. Manhattan focuses solely on distance resulting in inefficient exploration as walls and obstacles are not considered. To combat this, we introduced a weighting on obstacle cells in the global costmap and occupancy grid. Although well-performing, this is still considered a limitation as the solution does not match the robustness of path searching algorithms. A possible improvement could be to implement an optimal pathing algorithm in another language (e.g. c++) that can execute more efficiently during runtime.

ii) Pose estimation – existence of rotational distortion

As previously mentioned, the existence of point cloud distortion whilst moving at high velocities can cause large inaccuracies and can give multiple detections for a single instance. In response, we introduced a naïve approach to distortion reduction by averaging the estimation over a specified number of readings. Improvements were also made by decreasing velocity parameters and thresholding by its distance and angular velocity. A possible improvement could be to implement a distortion correction by anticipating the time delay [6].

iii) Computational power – high requirement for CNN, point cloud and local planner

Object detection using darknet proved difficult in terms of computational power. The requirement to process the stream of images in real time, resulted in noticeable lag. Similar performance issues were seen in the publish rate of point cloud topics and the control rate of the teb local planner. All these issues resulted in a significant decrease in system performance. Efficiency was improved through the use of cuDNN (a GPU-accelerated library), parameter-tuning, a reduction in model complexity (tiny-yolov3) and resolution. Aside from small efficiency improvements, this is mainly seen as a limitation based on the hardware. Unfortunately, due to the performance implications, we currently see cuDNN as a hard requirement, requiring an NVIDIA GPU. A possible improvement to our simulation that could prove useful for evaluating our approach is GPU integration with Gazebo. This would reduce some performance issues due to computational requirements and increase focus on the design.

IV. Personal Reflections

In conclusion, the generalised solution proposed is effective in a controlled Gazebo environment. Available sensor streams are well utilised to perform intelligent tasks to deal with unseen environments. Functionality has been included to extend the model to accommodate different environments, obstacles and objects. Better performance may be achieved with a less generalised solution (e.g. optimal waypoints set in known environments), however our solution remains competitive and the higher level of generalisation provides a greater value.

Future development could include the application of this approach on a real-world model. This could be useful as kinematics, dynamics and geometry could all differ from the simulation. Alternatively, improvements could be made to the existing design as discussed in the evaluation section.

Personal contributions: organised package structure, implemented and managed all xml files (launch, config, xacro etc.), integrated navigation stack, evaluation of local planners (base, dwa, teb), parameter tuning and optimisation, scripts to create and partition database, database labelling, darknet integration, model training/evaluation/improvements, data fusion for pose estimation (pair programming), distortion reduction (thresholding), creation of 10+ test worlds to evaluate detection and path planning performance.

Peer Evaluation

- Ryan Dave: 5
- Daanish Karim: 5
- Jack Parton: 5
- Michael Mee: 5

APPENDIX

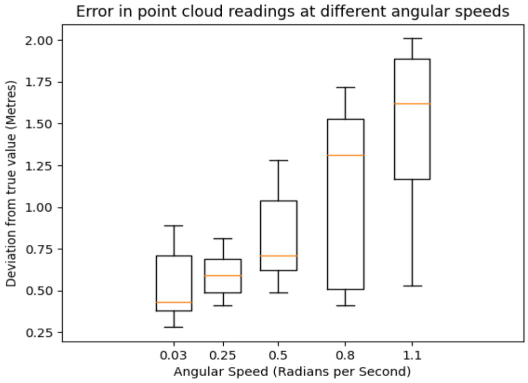

Figure 6. Box plot of pose estimation error for varying angular velocities

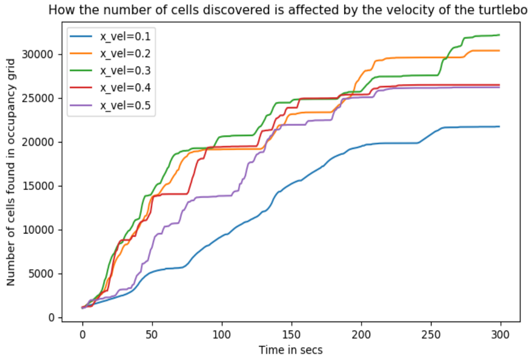

Figure 7. Line graph representing grid coverage over time for different values of x_vel

Predicted

		Positive	***Negative***
True	***Positive***	***53***	***11***
	Negative	***0***	***152***

Figure 8. Confusion matrix results from cross validation (6000 iterations/Green Box)

REFERENCES

[1] M. R Dogar, M. C Koval, A. Tallavajhula, S. Srinivasea, “Object search by manipulation”, Auton Robot vol. 36 pp. 153–167, 2014.

[2] B. Yamauchi, “A frontier-based approach for autonomous exploration”, 1997.

[3] M. Bjelonic, “Yolo ros: real-time object detection for ros", URL: https://github.com/leggedrobotics/darknet_ros, 2018.

[4] J. Redmon, S. Divvala, R. Girshick, A. Farhadi , “You only look once: unified, real-time object detection”, 2016.

[5] W. Pereira, G. Silva, O. Junior, K. Vivaldini, “An extended analysis on tuning the parameters of adaptice monte carlo localisation ros package”, The International Journal of Advanced Manufacturing Technology vol. 117 pp. 1975-1995, 2021.

[6] K. Zheng, “Ros navigation tuning guide”, 2016.

[7] T. Renzler, M. Stolz, M. Schratter, D. Watzenig., “Increased accuracy for fast moving lidars: correction of distorted point clouds”, 2020.