Deploying the Thorvald system using SLAM

Harry Rogers *School of Computer Science*
University of Lincoln, United Kingdom
https://orcid.org/0000-0003-3227-5677

***Abstract*—Robotic simultaneous localisation and mapping (SLAM) is an autonomous technique that allows systems to deploy into unknown areas and map them accurately. This paper presents a novel system architecture to deploy the Thorvald system in a static map to map the area as well as any potential weeds the system finds. The system architecture uses A* with move base to roam the map with a frontier detection input, the system utilises Gmapping and an Extended Kalman Filter to build an accurate map. It was found that the maps built were accurate and could be re used for other systems that are deployed in the area. Future work is discussed to improve the system using other methods. The package thor-explorer can be found at https://github.com/Harry-Rogers/thor-explorer**

***Index Terms*—SLAM, A*, ROS, Gmapping, Move base, Python, Frontiers**

I. Introduction

TRUE autonomous deployment in robotics is very difficult to ensure, maintain and create. The idea behind deployment is not only to put a robotic system in an unknown location and be able to react to its surroundings but it is also to ensure that these surroundings are mapped so that other systems can benefit from this information. The goal of this system is to map the surrounding area. The objectives are listed below:

- Map area without human intervention.
- Map any potential weeds that are spotted.
- Complete this in a suitable time frame.
- Avoid crashes.

To be able to complete the aims and objectives simultaneous localisation and mapping (SLAM) needs to be used as this will ensure that the map is built correctly.

A. SLAM

SLAM is an enormous subject that has many options. However, when using SLAM, the key ideas are to localise the robotic system inside of a map that is being built. With the system proposed, Extended Kalman Filter (EKF) is used for localisation alongside this the ROS Gmapping package, which will be explained later in section III, is used to map the area. SLAM allows for the system to build a map that is accurate and re-usable for other systems to go back to the area and complete other tasks.

The rest of the paper is as follows: Section II covers the Related work including methods of exploration. Section III details the System Architecture, which details how the package was created. Finally, Section IV evaluates the package, references are at the end of this paper.

II. Related work

There are abundant methods to be able to deploy robotics autonomously and build maps of the surrounding area. The best method for this is the overarching idea of simultaneous localisation and mapping (SLAM). To break down the problem of mapping there are two subsections, those being localisation and mapping. The act of mapping could use something like an occupancy grid to build a map by showing occupied and unoccupied cells. The act of localisation is trying to determine where inside this map is the robotic system. This paper will explain how Rapidly-exploring Random Trees (RRT) work as well as Sensor based Rapidly-exploring Random Trees (SRT) as these are some methods that could be used for figuring out whether an occupancy cell is occupied or empty. These will be explained as originally this paper was going to use these methods. Then the frontiers algorithm will be analysed which was like the actual algorithm that was used in the system architecture.

A. RRT

Rapidly-exploring Random Trees is an algorithm that uses trees to create an exploration type algorithm. The concept takes a random point and using the robots' position will branch away from the robot towards the random point. The length of the branch will be set to a specific distance that the system can drive in one motion. The branch will not reach the random point on the first iteration of the algorithm so more branches can be made to explore the area. A branch is only continued when it is the closest known point to the random point it is looking at exploring. Once a new point has been made this will be considered a node therefore there will be branches with nodes on them. The algorithm can have an end goal which could be coordinates of an unoccupied cell using an occupancy grid. When RRT has an end goal, periodically, the random point will be assigned to be the end goal. This would mean that RRT could use frontier exploration to explore and build maps. This algorithm was used in [1] that has a built package for ROS called rrt_exploration. However, this algorithm is computationally expensive compared to what is used in this paper. Move base that was created using ROS [2] is used in this paper due to the reasons explained in section III. Developing more on this RRT idea SRT could be used for dynamic maps.

B. SRT

Sensor based Rapidly-exploring Random Trees uses most of the same concepts as RRT. The significant difference between them is the use of a single branch over multiple branches. This means that the robotic system will drive to the new node it has created whilst checking there are no obstacles. If there is an obstacle SRT will go back to the last known node and branch from there. The paper [3] explores this and creates other similar methods with SRT-star and SRT-Ball. Despite this, SRT is similarly to RRT computationally expensive compared to using move base. SRT also has a lot of trouble with backtracking and potentially going to places that have already been explored. This paper [4] does improve this issue and states that there is an improved run time of up to 30%. However, this still is not as effective as what the method this paper uses.

C. *Frontier algorithm*

The original paper for frontier-based exploration [5]. Describes the idea of looking for frontiers in occupancy grids, frontiers are cells that are unknown in an occupancy grid. If a cell is unknown to be able to map an area correctly the cell either needs to be occupied or known to be empty. The approach uses a laser scanner to detect points in the map that are either occupied, unoccupied or empty. This approach is very intuitive as it makes sense for a system to move to a place it has not seen to build up the map. Despite this, if a robot system were to move to that exact spot that is unknown would not the system be able to detect the frontier before it arrives at the frontier with the use of the laser scanner range. This is the main difference between the original frontiers paper and this paper. This paper suggests going near the frontier to decide whether it is an occupied cell or empty. This would allow for a shorter execution time, with less time spent going to exact locations when this is not necessary for map building when the map can be explored from a distance.

III. System architecture

The system architecture begins with launching Gmapping, EKF, Move base, Rviz and then 4 nodes created in Python. These being the shutdown sequence, weed detection, weed publisher and frontier mapping. The frontier mapping script will find coordinates it is interested in exploring, it will then pass these to move base to go to. Whilst this is happening the weed detection and publisher are looking for weeds and publishing them to the Rviz to be visualised as a point cloud. The frontier mapping script will continue until the shutdown sequence begins. When the shutdown sequence begins, this script will find all nodes created and shut them down. Once this is complete it shuts down itself. The key points of the architecture are explained below.

A. *Frontier detection*

To detect places that should be explored a frontier detection script was created. It is very similar to what is used in the rrt_exploration [1] package which reads in the occupancy grid data. Then using OpenCV the walls of the map are found using functions like contours and canny edge detection. Once this is completed the coordinates that are found are sent to move base, the array of values is then sorted in the move base file. The system uses the last point found in the array for the first point to go to as this is deemed to be one of the furthest points away. Once this is selected the value is changed by 1 metre, this will be explained later. Once the system has arrived at the point it is interested in, the system will scan again looking for points that have not been explored and will then go to those points using the same method. The system finishes when there are no more points to explore, or the system will finish once it has been to three points as this is deemed to be enough for the map to be complete. The shutdown sequence is started. Shown below in figure 1 is the first iteration on the static map for frontier detection before calculating the coordinates for the frontier chosen.

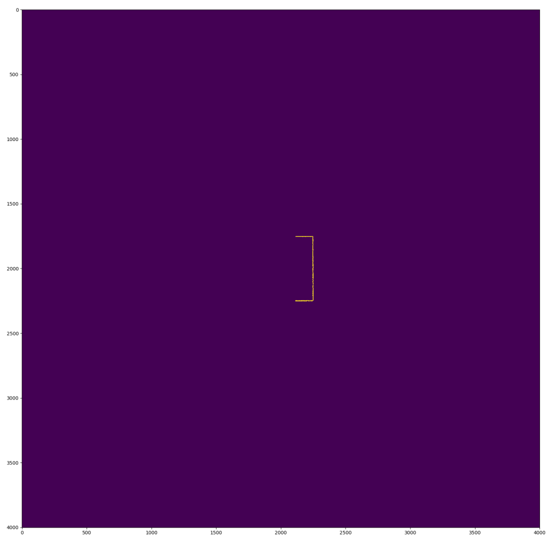

Fig. 1. Frontier detector on first iteration.

B. *Move base*

Move base in this system is key as it allows for the map to be built. The planner file for move base is setup so that the default global planner is used navfn/NavfnROS, this has two parameters set these being True for visualize_potential and False for use_dijkstra. This allows for A* to be used for global planning which can be faster than the default of Djikstra's path planning. The base local planner used is the DWAPlannerROS, this has two parameters also that are holonomic_robot which is False as the Thorvald system is not holonomic, the other parameter set is xy_goal_tolerance which is set to 2.0 metres.

Combining the above planners and the aforementioned change to the coordinates allows for the system to be able to explore the map at a much more rapid pace. The system has a laser scanner which has a large range; therefore, it was deemed that it was not necessary to get to the exact frontier that needs to be explored as the occupancy cell will have been determined way before the system actually arrives at the frontier it originally detected. This allows for the system to get to point faster and explore more of the map without having to try and get to an exact point.

C. Mapping

To be able to map the surrounding area the Gmapping package from ROS was deployed. There are many types of mapping packages that offer an occupancy grid and a method of mapping. Other packages like HectorSLAM could have been used but this package does not utilise every aspect of the Thorvald system, more specifically it does not use the odometry from the system. The paper An Evaluation of 2D SLAM Techniques Available in Robot Operating System [6] reports that the most robust option would have been Gmapping. This is because of the low CPU usage alongside this Gmapping had the lowest error estimation for both simulation tests that were completed. 3D mapping techniques will be discussed in future work as these will greatly improve what can be shown in a map.

D. Open CV weed detection

OpenCV is used heavily in this system, it has two main uses those being frontier detection and weed detection. When looking for weeds it creates a median blur to begin with, this is to create blobs in the image, so it becomes easier to detect weeds. Using the hue saturation and intensity values (HSV) a colour threshold mask is used to identify the weeds in the image. The next step uses dilation to make any blobs that are detected bigger and easier to see. Finally, a blob detection implementation from OpenCV is used to be able to distinguish between weeds and crops. Key points are drawn and converted to be able to publish the points in a weed map. Shown below in figure 2 is the original image with the key points image.

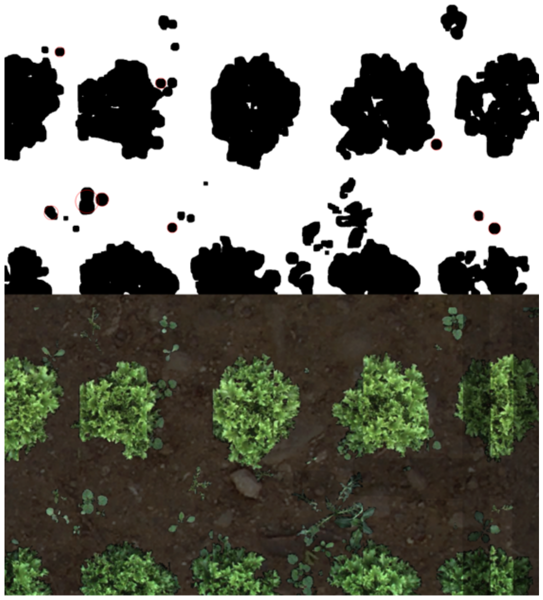
Fig. 2. Weed detection showing key-points and original image.

E. End sequence

The end sequence involves all the above information. Move base is sent to two distinct locations, the origin point (0,0) and a pseudo random point between -5 and -10. These points are to explore the map and find weeds to publish to the map. Once the Thorvald system has reached the second point a map is made using the map_server as this will save the current map created. This map is stored in the packages map directory. This sequence will then shutdown all nodes apart from rviz and gazebo, this is to ensure that the weed map is saved, and everything is ready to be shut down.

IV. Evaluation

Overall, the system is effective at mapping the area and creating a weed map. The system does fulfil the objectives set out in section I. The system maps the area without intervention to stop the Thorvald system from harm and the system avoids physical crashes. The system completes mapping in a suitable time as this is usually around 5 minutes. Alongside these aims it also maps potential weeds that are seen from the system. The system works currently in the static map and it is very clear to be able to see where potential weeds are spotted. Shown below in figure 3 is a perfect version of the static map with weeds on it created from this package, this was created by letting the system roam around for a significant time unsupervised.

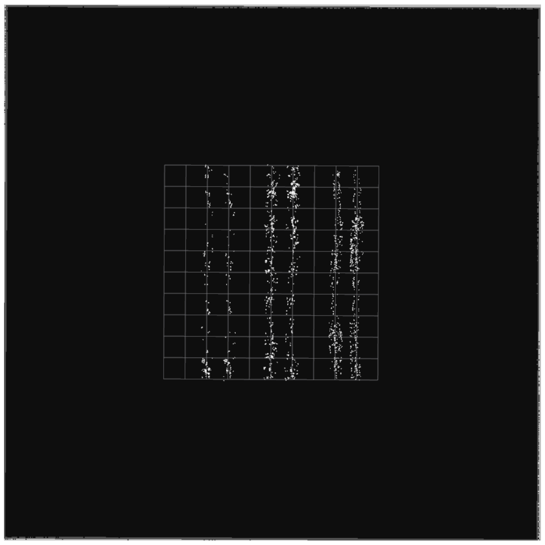
Fig. 3. Static weed map.

The system deployed will map the area fully, however the weed map created will not be fully complete until all weeds have been mapped. This becomes problematic for trying to find all the weeds, this is partially solved by the use of the end sequence. However, this is not full autonomy as there is no thinking from the system other than to go to these specific places. These were chosen to show that the system can complete the task it has set out to do. Another task that was part of this, but not detailed in the aims, was to try to spray weeds with the sprayer when weeds are found. However, the sprayer releases a large square blob that can cover not only the weed but also the crop. Therefore, this system only deploys the sprayer when it sees a weed and has calculated that it the sprayer is now near the weed.

A. Future work

However, no system is perfect and nor is this one. There are some shortcomings with this system, when using the dynamic map, it becomes difficult to map the area correctly. This is due to some of the obstacles in the map. These being the car wheel, pallet and tree roots. Those obstacles are below the laser scanner therefore the system cannot see them so it is very likely that the system will crash into them. When the system crashes into something in gazebo that it cannot see it is very likely to break the simulation. The only time this does not happen with the obstacles mentioned is the tree roots, this is because the system begins mounting the roots and becomes stuck in motion. Despite this, there is a bigger problem of the actor. The actor walks around taking the same path and frequently walks through the Thorvald system breaking the simulation in the process. One of the main issues with detecting the actor is when the Thorvald system is in the actors' path and facing a different direction so that the system cannot see the actor. This frequently happens with move base as the system moves very slowly, so the Thorvald system will usually appear in this scenario. However, even if all these conditions were completed by lowering the laser scanner and either removing the actor or navigating around them. The system does not display weeds well with obstacles on the map. Shown in figure 4 is what the dynamic map looks like if you lower the laser scanner and ignore the actor.

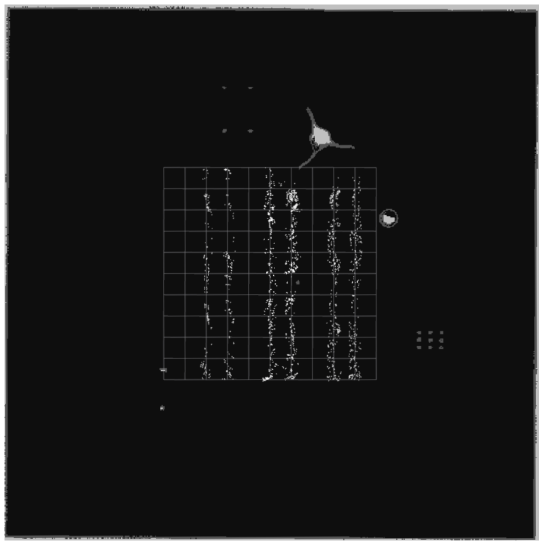

Fig. 4. Dynamic weed map with laser scanner lowered.

The issue with this is that it is hard to differentiate between the telephone pole, car wheel, pallet and dumpster that is in the middle of the middle crop row and weeds around it. This would mean that a different weed mapping technique would need to be used, something like a topological map could work. It would not be possible to use an occupancy grid as this again would cause confusion between smaller objects and weeds. Thankfully the weed map array that is implemented can be saved to a csv file so that the coordinates of potential weeds can be used later by other systems that could identify weeds at those coordinates.

To be able to deploy this system the real world this system would need to integrate the use of Adaptive Monte Carlo Localisation (AMCL). The use of EKF allows for the localisation to be far better than it would be in real life if AMCL was used, this in simulation is used to be able to show the optimal use case for the mapping side of the system.

To better optimise the weeding portion of the system the introduction of a better spraying method could be used. For example, a robotic arm that sprays precisely weeds would be an excellent addition to the Thorvald system.

The weed detection in the system proposed can detect weeds in the map however, this could be improved with a better system using OpenCV or it could be completely replaced by a neural network that could detect weeds.

REFERENCES

[1] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1396–1402.

[2] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," vol. 3, 01 2009.

[3] G. Oriolo, M. Vendittelli, L. Freda, and G. Troso, "The srt method: randomized strategies for exploration," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 5, 2004, pp. 4688–4694 Vol.5.

[4] H. El-Hussieny, S. Assal, and M. Abdellatif, "Improved backtracking algorithm for efficient sensor-based random tree exploration," 06 2013.

[5] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE, 1997, pp. 146–151.

[6] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2d slam techniques available in robot operating system," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2013, pp. 1–6.