Autonomous Exploration and Mapping using a Mobile Robot Running ROS

Rachael N. Darmanin and Marvin Bugeja
Department of Systems and Control Engineering, University of Malta, Msida, Malta

Keywords: Wheeled Mobile Robots, ROS, Autonomous Exploration, SLAM.

Abstract: This paper proposes a practical solution to the autonomous exploration and mapping problem using a single mobile robot. Moreover, the authors implement the proposed scheme within the Robot Operating System (ROS), and validate it experimentally using PowerBot, a real wheeled mobile robot equipped with a 2D laser scanner. In essence, the proposed scheme integrates an efficient particle-filter-based SLAM algorithm, two different exploration strategies, and a path-planning and navigation module. The modular nature of the proposed scheme is intentional and advantageous. It allows the authors to compare the two exploration strategies under investigation objectively and with ease. Finally, hypotheses testing is also used to strengthen the results of the comparative analysis.

1 INTRODUCTION

Exploration and mapping by an autonomous agent have been of particular interest to the robotics community for decades. Research shows that the autonomous mapping of an unknown environment by a mobile robot is a non-trivial task (Thrun et al., 2005). Global Positioning Systems (GPS) and the use of beacons for mapping an unknown environment may facilitate the task of mapping (Dissanayake et al., 2011). However, such systems suffer from a number of inherent limitations, namely, the inaccessibility of GPS indoors and the laborious setting up of strategically placed beacons. Hence, the concept of *Simultaneous Localization and Mapping* (*SLAM*), was developed in order to amalgamate sensory data into a world model (or a map). Through probabilistic techniques, SLAM algorithms are capable of processing sensory data to estimate the location of a robotic system and a map of its environment at the same time.

In robotics, maps are mainly acquired to enable autonomous navigation. However, mapping using robots can also be useful when it comes to explore and map dangerous environments, or places that are inaccessible to human beings. Hence, *autonomous exploration strategies* were developed, which enable the robot to determine the next location to explore in order to improve its map and localization estimates. Exploration strategies place an emphasis on autonomously mapping as much of an unknown environment as possible, and in the shortest time possible. Other researchers argue that rather than just *exploratory* actions, accurate mapping requires *exploitation* actions, such as place revisiting actions (Makarenko et al., 2002). This gave rise to *Active SLAM* strategies, that seek to improve the localization estimate of the robot rather than explore as much of the environment as possible in the shortest time. However, both *exploration* and *Active SLAM* strategies ultimately provide the robot with a sequence of locations that it needs to visit in order to meet the specified exploration criteria.

For a robot to autonomously explore and map an environment, it must follow a set of navigation instructions that allow it to proceed from one location to the next safely, even in the presence of obstacles. During the years, several *path planning and motion control* algorithms have been proposed which enable the robot to move to the chosen location. The integration of a SLAM algorithm, an exploration or Active SLAM strategy and a path planning/navigation algorithm allows the robot to autonomously explore and map an unknown environment using just sensory data.

The purpose of this work was to develop a single robot system, running Robot Operating System (ROS), that can autonomously explore and map unknown environments efficiently. The system needed to be modular, in that the SLAM, path planning and motion control modules, can be used with different exploration strategies, without any modifications.

Darmanin, R. and Bugeja, M.
Autonomous Exploration and Mapping using a Mobile Robot Running ROS.
DOI: 10.5220/0005962102080215
In *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2016) - Volume 2*, pages 208-215
ISBN: 978-989-758-198-4

ROS facilitates this modular implementation since it allows a set of independent processes to communicate through a message passing structure.

The main contributions of this work are: 1) the development and experimental evaluation of a new ROS package consisting of two autonomous exploration strategies for mobile robots. To the best of the authors' knowledge, no other ROS package offering the same functionality is available to date; 2) a comparative study of the two mentioned exploration strategies, based on physical experiments in a real-life environment.

The rest of this paper is structured as follows. Section 2 provides a review of the existing autonomous exploration and mapping systems. Section 3 describes the main hardware used in this implementation and the underlying software framework that makes up the whole robotic system. Section 4, presents a set of experimental results while Section 5 contains a comparative analysis, backed up by statistical hypothesis testing, of the two exploration strategies under investigation. Finally, conclusions from this work are drawn in Section 6.

2 RELATED WORK

Being aware of one's environment means that: one is able to perceive the environment, localize himself in it, and be able to navigate effectively in this environment. In the field of mobile robotics, this involves learning maps of an unknown environment, which is the integration of three non-trivial tasks that must be solved concurrently (Stachniss et al., 2005). Figure 1 illustrates this concept as the integration of mapping, localization and path planning and motion control.

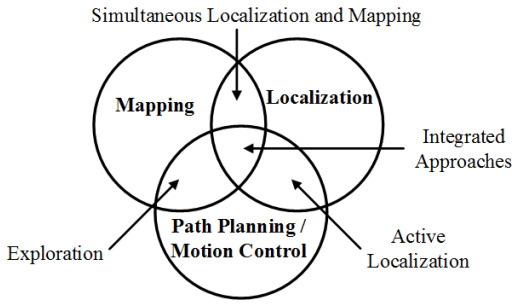


Figure 1: Illustration of the integrated approaches, adapted from (Makarenko et al., 2002).

Given a known map, several path planning algorithms such as Dijkstra's algorithm (Dijkstra, 1959), A* algorithm (Dechter and Pearl, 1985), D* algorithm (Stentz, 1994) and Dynamic Window Approach (DWA) (Fox et al., 1997), among others, have been proposed to allow a robot to navigate autonomously in the environment. During the years, there have been many advancements in the field of path planning algorithms that can be used seamlessly within the integrated problem of autonomous exploration and mapping. Moreover, the Dijkstra, A* and D* algorithms are all *global path planners*, in that they plan a path from the current robot location to the destination based on the current global map (the whole map acquired so far). The DWA was designed to make use of the velocity control space to search for control commands that must be sent to the robot in order to reach a goal destination. Hence, the DWA does not make use of a global map, but rather it uses the local map obtained through sensory data.

Perhaps the most complex problem that must be solved in the task illustrated in Figure 1, is SLAM. In autonomous exploration, the map estimate produced by SLAM is used by the exploration algorithms in order to identify which is the best location that the robot must visit next. Overall, SLAM algorithms may be organized into three main categories as follows: *Extended Kalman Filter (EKF) SLAM*, *Particle Filter SLAM*, and *Graph Based SLAM*. EKF SLAM was for many years the preferred choice since it requires a simple and straight-forward implementation and performs well in small areas having a limited number of landmarks. However, since EKF SLAM bases its estimate on just one hypothesis and assumes that the posterior is normally distributed, it may become overconfident in its estimate, leading to less accurate mapping (Dissanayake et al., 2011). Hence, Particle Filtering became the preferred probabilistic technique for many researchers. Particularly, in (Thrun et al., 2004), the authors integrate the common *Sequential Importance Resampling*, (*SIR*) filter with EKF to produce FastSLAM, which results in a faster algorithm, when compared to the common SIR filter on its own. Furthermore, Rao-Blackwellized Particle Filters are more commonly used to solve the SLAM problem since they attempt to reduce the *degeneracy* and *particle depletion* problems presented by less advanced particle filters such as the *Sequential Importance Sampling* (SIS) and SIR filters (Grisetti et al., 2007). In (Grisetti et al., 2007) the authors propose two improved techniques for Rao-Blackwellized Particle Filter SLAM, that make the algorithm more efficient and accurate in estimating the most likely pose as well as having a reduction in the particle depletion problem. The implementation of this improved Rao-Blackwellized Particle Filter is often referred to as GMapping, which is directly accessible online from `openslam.org`. Moreover, the use of Graph-Based

techniques in SLAM is fast gaining popularity. In Graph-Based SLAM, each node in the graph represents an estimated robot pose or the estimated position of a landmark, while the edges connecting these nodes encode in them sensor observations that are constrained by the estimated robot pose. The performance of Graph-Based SLAM is said to be comparable to that of Particle Filter-Based SLAM algorithms such as GMapping (Burgard et al., 2009).

Although the task of choosing the next location that the robot should visit seems to be intuitive, several researchers have developed several different exploration and Active SLAM algorithms for this purpose. One very popular and simple exploration strategy is the *Nearest Frontier* approach (Yamauchi, 1997). This algorithm simply analyses an *Occupancy Grid Map* and detects all potential borders (called candidate frontiers) between the cells marked as *OPEN* (free from obstacles) and those marked as *UNKNOWN* (not yet explored). Such a map is segmented into cells, where each cell is assigned with a probability of occupancy, with zero probability meaning that it is free from obstacles, and with a probability of one, meaning that it is *OCCUPIED*. Normally, a probability of 0.5 means that there is no information about the occupancy of the cell, and hence it is still *UNKNOWN* (Yamauchi, 1997). The distance between the current robot pose and each frontier, is analysed, and the frontier closest to the robot is chosen as the next location to explore.

Another algorithm that is designed to explore as much of the environment as possible is the *Next Best View* (*NBV*) approach proposed in (Gonzalez-Baños and Latombe, 2002). In this case, the evaluation criteria of the candidate destinations, attempt to strike a balance between the amount of unknown area that the robot can explore from that location, and the distance that the robot must travel to arrive there.

Unlike the Nearest Frontier and the NBV strategies just described, in (Makarenko et al., 2002) and (Stachniss et al., 2005), among others, the authors propose techniques that do not only focus on exploring as much of the environment as possible, but in addition aim to improve the localization estimate of the robot (Active SLAM). Particularly, in (Stachniss et al., 2005), the authors propose a technique that makes use of the entropy of a Rao-Blackwellized Particle Filter together with the distance from the current robot pose in order to evaluate candidate destinations.

To date, the robotics community is still in debate on which of these exploration strategies is best to solve the problem of autonomous exploration and mapping. To this end, several works such as those presented in (Amigoni, 2008; Carlone et al., 2014), among others, seek to compare such strategies. In (Amigoni, 2008), the author shows that the simpler algorithms, such as the NBV approach, yield the best results in the shortest time possible, and the robot travelling the shortest distance. However, in (Carlone et al., 2014), the authors show that when the focus is placed on the quality of the map, then those strategies that involve Active SLAM produce the best results, even if not in the shortest time possible. Such works indicate that the choice of exploration or Active SLAM algorithm is highly dependent on the application and on the criteria that shall be used in the evaluation of the results.

3 SYSTEM DESIGN AND IMPLEMENTATION

The concept of autonomous exploration and mapping represented in Figure 1 was implemented in ROS on PowerBot^{TM}, a differentially-driven wheeled mobile robot designed and manufactured by *Adept Mobilerobots* for research and rapid prototyping. PowerBot is equipped with a SICK LMS200 laser scanner, an Advanced Robotics Control and Operations (ARCOS) robot controller for low-level motion control, as well as an on-board computerfor algorithm development. The on-board computer is interfaced with the robot controller via Advanced Robotics Interface for Applications (ARIA), which is a library of functions that are capable of handling the lowest level details of the client-server interactions, such as controlling the speed of the robot.

Robot Operating System (ROS) is a software framework that is ideal for the development of robotic applications. It consists of a number of software packages that are used to perform particular tasks, for example, SLAM. ROS facilitates modular implementation of different functions, in that processes that are currently running on the system (called *nodes*), interact between themselves through a message passing system, as shown in Figure 2. Each packet of data is sent between nodes according to the *topic* that it has been assigned. The topic defines the type of data message that is sent. Furthermore, a node may either be a *publishing node* (broadcasting data) or a *subscriber node* (receiving data). Hence any node subscribing to the same topic shall receive the same data being broadcast by the publisher. For instance, in Figure 2 the publishing node “talker” is communicating with the subscriber node “listener” by sending data messages over the topic “chatter”. These interactions are controlled by the master node called “roscore”.

The modularity of ROS leads to a rather simple

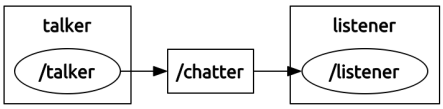


Figure 2: ROS architecture of nodes and topics.

design that emerges from the integration of the fundamental concepts and the system block diagram depicted in Figure 3. Going from the block diagram in Figure 3 to a system designed in ROS requires the identification of the specific package to be used in place of each component in the block diagram. The parameters of each package must then be tuned for the specific robot platform and application.

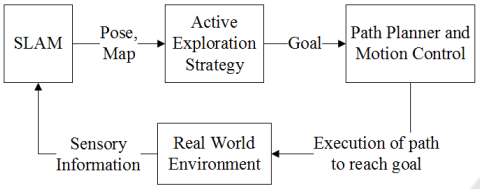


Figure 3: ROS architecture of nodes and topics.

To perform SLAM, the gmapping package, which implements the Rao-Blackwellized particle-filter-based approach developed in (Grisetti et al., 2007), was chosen due to its ability to map an environment in an accurate and efficient manner. In contrast to EKF-based SLAM, this technique uses multiple hypotheses represented by the particle set in the particle filter. The node, called "slam_gmapping", subscribes to the raw odometry data and laser scan data over the "/RosAria/pose" and "/scan" topics, respectively. The package "ROSARIA" acts as a wrapper for the ARIA libraries in ROS. It allows the transmission of high level commands to the robot controller from ROS. In fact, the odometric pose estimate of the robot is published by the "RosAria" node. In turn, the "slam_gmapping" node publishes the map data, which is then used by the exploration algorithm to compute the next location that the robot needs to explore.

The published map is an *Occupancy Grid Map* vector. Moreover, some of the "slam_gmapping" parameters had to be tuned in order to extract the best performance in the given environment with the given robot. For example: two particular parameters were used to enable the SLAM package to execute a scan every time the robot translates by 0.2 metres or rotates by 0.1 radians. The settings were chosen in an attempt to obtain the best map with the available computational power, as more frequent scans naturally require more computational power. Furthermore, the laser range was also tuned in accordance with the specifications of the laser scanner, namely the maximum range setting was set to five metres. Finally, the map was set to be updated every three seconds.

Although there are some packages that implement the *Nearest Frontier* exploration approach in ROS, these presented certain limitations, such as not being fully autonomous, and thus could not be used for the intended purpose. Hence, the *Nearest Frontier* approach proposed in (Yamauchi, 1997) and the *Next Best View* approach proposed in (Gonzalez-Baños and Latombe, 2002) were both implemented from scratch as ROS nodes in a new package called "exploration". To the best of the authors knowledge, this is the first time that these two algorithms are being made available in ROS.

For both exploration strategies, the potential candidates for exploration emerge through the use of frontier detection which was also proposed in (Yamauchi, 1997). Frontier detection is performed in two steps:

1. The search for frontier cells, and
2. The grouping of said frontier cells into frontiers comprising adjacent frontier cells only.

In the search of frontier cells, the algorithm traverses the map data and creates an vector of frontier cells. These frontier cells are then grouped into frontiers according to their adjacency. Each frontier is then stored in a two-dimensional vector which is used by the exploration algorithm for further analysis. Eventually, the exploration process terminates when the frontier detection algorithm can no longer detect frontiers that are at least ten cells long. This applies to both exploration approaches that were examined in this work.

The difference among the two approaches lies in how they evaluate the candidate destinations and choose the location that the robot must explore next. In the *Nearest Frontier* approach, a global path is planned from the current robot pose to each candidate. The candidate that is closest to the robot (i.e. the one having the shortest path) is then chosen. On the other hand, the *Next Best View* approach implements the following equation to rank the potential candidates,

$$g(q) = A(q)exp(-\lambda L(q,q_k)) \quad (1)$$

where $g(q)$ represents the ranking function, $A(q)$ is the total unknown area that is expected to be covered by the laser scanner from a candidate location, and $L(q,q_k)$ is the length of the path planned from the current robot pose to each candidate location, using the global occupancy grid map published by the SLAM algorithm.

The path length, $L(q,q_k)$ is computed in the same manner as it is computed in the *Nearest Frontier* approach. This means that the same global path planner used to plan accessible paths for navigation is used to

plan a path for the robot in an incomplete occupancy grid map. The length of this path is then computed. The unknown area that the robot can potentially "see" from a candidate location, $A(q)$, is calculated by a ray casting technique, which is also used in (Gonzalez-Baños and Latombe, 2002). Ray casting attempts to simulate the data obtained through a single scan of the laser scanner. Given a scan resolution, the maximum laser range and the scanning angle of the real laser scanner, a number of rays are projected from a candidate location. If the ray encounters a known cell (*OPEN* or *OCCUPIED*), then the length of the ray from that point to the maximum ray length is cropped out. Consequently, the endpoint of the ray is the point at which the ray hits a known cell, and therefore, the length of that ray is the Euclidean distance between the candidate location coordinates and the coordinates of the ray endpoint. Thus, the unknown area that may be visible from a candidate point is the summation of the lengths of the rays projected outwards over the unknown cells only, as seen in (2), where (x_c, y_c) are the map coordinates of the candidate location and (x_{end}, y_{end}) are the coordinates of the ray endpoint.

$$A(q) = \sum_{No.\,of\,rays} \sqrt{(x_{end} - x_c)^2 + (y_{end} - y_c)^2} \quad (2)$$

Furthermore, the parameter λ plays an important role in the selection of the next best view since it allows the robot to either prefer shorter routes over a larger area of visibility, or vice-versa. If the value of λ is small, the algorithm prefers the candidates from which a larger unknown area may be explored and hence gives them a higher rank. If λ is large, the algorithm gives a higher rank to those candidates that are closer to the robot. Eventually the candidate that has the largest ranking value, $g(q)$, is chosen as the next location that the robot must navigate to and explore.

As soon as the next destination is chosen, the robot must navigate to that location. To do so, path planning and motion control algorithms are required to allow the robot to navigate effectively in an environment cluttered with both static and dynamic obstacles. For this purpose, the package "move_base" was used. This package requires the pose of the robot within the map to be able to plan a global path from the current robot pose to the goal destination. In order to take the dimensions of the robot into consideration, the path is planned in a global costmap that inflates the cost around each *OCCUPIED* cell. If the robot traverses the inflated circle around such a cell, it may be either close to colliding with an obstacle or in fact, it may be in collision, depending on the cost of each cell. This global costmap is constructed on the static occupancy grid map that is being built by SLAM. Moreover, having just a global path planned does not enable the robot to avoid dynamic obstacles such as people walking around it. To mitigate this problem, the package makes use of a local path planner, which plans intermediate paths using a local map. This local map is another costmap which is constructed using the laser scan data directly. Therefore, if a dynamic obstacle appears in the local path planned of the robot, this would be detected by the laser scanner and the local path planner changes course accordingly. However, the local path planner always seeks to guide the robot as close to the global path as possible. For this effect, the global and local costmap parameters were tuned to the dimensions of the robot. Furthermore, the global path planner adopted in this case is the Dijkstra's Algorithm (Dijkstra, 1959), while the local path planner adopted is the Dynamic Window Approach (Fox et al., 1997). The implementation of these path planners were both available in ROS.

4 RESULTS

One of the aims of this study was to compare the mapping accuracy of two *exploration* strategies which were implemented as part of a complete autonomous exploration and mapping system in ROS. The first set of results presented in Section 4.1 shows how the different exploration strategies choose different exploratory locations from a set of candidates. Furthermore, Section 4.2 presents a set of results obtained through a number of real life autonomous exploration and mapping experiments. The map obtained in each experiment was compared to a ground truth map, illustrated in Figure 4, which was obtained by manually steering the robot in the environment. Hence, the *Acceptance Index* metric could be calculated. This comparative metric was introduced in (Carpin, 2008). The acceptance index, ω, is calculated as the ratio of the agreement between the known cells of a ground truth map, $M1$, and a resulting map, $M2$, to the total sum of agreed and disagreed cells, as shown in (3). When $\omega = 0$ it means that the two maps are completely distinct and when $\omega = 1$ it means that the two maps are completely identical.

$$\omega(M1, M2) = \frac{\mathrm{agr}(M1, M2)}{\mathrm{agr}(M1, M2) + \mathrm{dis}(M1, M2)} \quad (3)$$

4.1 Case Study

In this case study, an incomplete map of the University of Malta Faculty of Engineering ground floor was

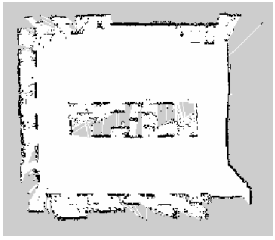


Figure 4: Ground truth map obtained by manually steering the robot in the laboratory environment.

used, where three candidate locations were detected by the frontier detection technique as shown in Figure 5. Candidate 3 shows a frontier that is made up of a collection of connected smaller frontiers. This is due to the diffused laser rays in that area which resulted in several gaps of unknown cells between the free cells. For this scenario, the ranking values of each approach are tabulated in Table 1. In order to show how the NBV approach can be used to rank candidates differently, the parameter λ was set to four different values from the set $\{0.02, 0.15, 0.5, 1.0\}$.

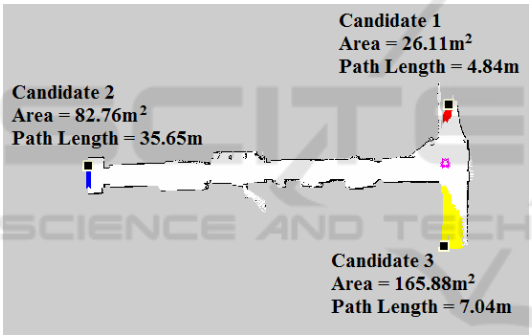


Figure 5: Incomplete map of the University of Malta Faculty of Engineering ground floor showing three exploratory candidates, together with the respective path length (in metres) from the current robot pose and the expected unknown area (in metres squared) that is visible from each candidate.

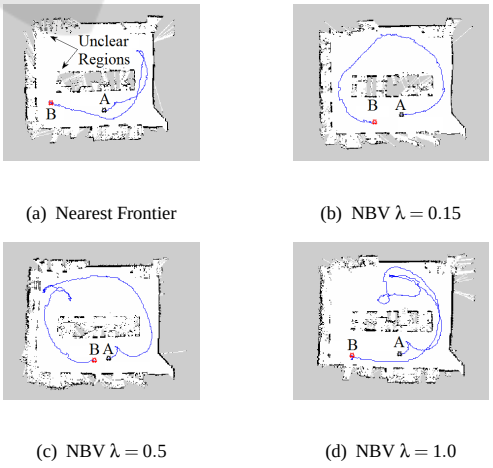
Table 1: Case Study results showing the value of the ranking function for each candidate, $\{1, 2, 3\}$, where the ranking function of the Nearest Frontier is the path length, which must be *minimized*, and that of the NBV is $g(q)$ as shown in (1), which must be *maximized*. Note that the ranking value of the chosen candidate for each approach is italicized.

Approach	1	2	3
Nearest Frontier	*4.84*	35.65	7.04
NBV $\lambda = 0.02$	23.70	40.55	*144.1*
NBV $\lambda = 0.15$	12.64	0.39	*57.74*
NBV $\lambda = 0.5$	2.33	1.49×10^{-6}	*4.92*
NBV $\lambda = 1.0$	*0.21*	2.67×10^{-14}	0.15

As can be seen in Table 1, the Nearest Frontier approach chose Candidate 1 as the next exploratory location since it offered the smallest path length. The algorithm of interest is NBV. When the value of λ was small (i.e. $\{0.02, 0.15, 0.5\}$), Candidate 3 was always preferred since it offered the largest area of visibility even though it was not the closest location to the robot. When λ was equal to 1.0, the algorithm preferred Candidate 1 since it is the location closest to the robot. Hence, one can conclude that as the value of λ increases, the NBV approach starts preferring candidates that are closer to the current robot pose and thus, this approach becomes similar to the Nearest Frontier approach.

4.2 Completely Autonomous Exploration and Mapping Results

In order to validate and compare the algorithms, the whole system was used in a set of four experiments where for each experiment, seven trials were recorded. In each experiment, the robot autonomously explored, navigated and mapped a controlled environment (a laboratory) consisting of several static obstacles such as desks and chairs. In the first experiment the *Nearest Frontier* approach was used while in the other three experiments, the *NBV* approach was used with different values of $\lambda = \{0.15, 0.5, 1.0\}$. These values of λ were chosen heuristically depending on the visual inspection of the robot behaviour. Figure 6 illustrates a typical map resulting from one trial for each of the adopted approaches while Table 2 presents the average (across the seven trials in each experiment) acceptance index, distance travelled and total exploration time for each approach.



(a) Nearest Frontier (b) NBV $\lambda = 0.15$

(c) NBV $\lambda = 0.5$ (d) NBV $\lambda = 1.0$

Figure 6: Sample map results for each approach showing the robot trajectory. 'A' and 'B' indicate the initial and final robot poses respectively.

Table 2: Autonomous exploration and mapping results showing the average acceptance index, average distance travelled (metres) and the average exploration time for each approach (seconds).

	Acceptance Index	**Distance Travelled**	**Exploration Time**
Nearest Frontier	0.82	25.41	132.48
NBV $\lambda = 0.15$	0.86	30.95	156.28
NBV $\lambda = 0.5$	0.84	40.44	140.90
NBV $\lambda = 1.0$	0.83	35.15	160.56

Visual inspection of the resulting maps, compared to the ground truth map in Figure 4, and trajectories, reveals that the Nearest Frontier approach keeps the robot from visiting certain locations that are quite far from its current pose. This then leads to unclear features and regions in the map, as shown in Figure 6a. On the other hand, in general, with the NBV approach the robot seems to cover a larger portion of the environment, hence leading to more accurate maps. It is important to note that the two approaches are subject to the same termination condition, specified in Section 3, to end exploration. Furthermore, in this study, although the focus is placed on the map accuracy, the distance travelled by the robot and the total time of exploration were also recorded to analyse the efficiency of the algorithms.

5 DISCUSSION

A question arises, of whether it is better to use a simple and fast exploratory algorithm like the Nearest Frontier (NF) approach, or the more complex and less time-efficient Next Best View (NBV) approach. To evaluate the map accuracy objectively, the differences between the acceptance indices of the seven trials for each approach were statistically tested. The boxplot in Figure 7 illustrates the distribution of these indices for each of the four adopted approaches. Intuitively, Figure 7 indicates that the best accuracy was obtained by the NBV approach since this exhibited the highest mean and the smallest variance. However, as the value of λ is increased, the NBV approach starts converging towards the NF approach. This intuition was verified statistically by using the One-Way ANOVA hypothesis test, where the null (H_0) and alternative (H_1) hypotheses were formed as shown in Table 3.

The main result of ANOVA reported that there exists a *significant* difference in the acceptance index performance of the four experiments under test. Hence, this means that H_0 was rejected. However, this test does not reveal *where* the significant difference occurs, and hence the Tukey HSD *post-hoc* test was performed. The outcome of this test shows that while there is a significant difference between the NF algorithm and the NBV algorithm with $\lambda = 0.15$, there is no significant difference among the other groups. Since the mean of NBV with $\lambda = 0.15$ is larger than that of NF, then the NBV approach with $\lambda = 0.15$ is significantly *better* than NF. Furthermore, the lack of significant difference between NF and NBV with $\lambda = 0.5$, and NF and NBV with $\lambda = 1.0$ indicates that for higher value of λ the NBV algorithm converges to the NF algorithm. One may argue that this is in line with the theoretical expectations since as λ increases, the NBV approach prefers the closer candidates (i.e. shorter path lengths) over candidates that offer a higher area of visibility. Furthermore, one must note that the parameter λ is application-specific. Thus, the threshold value at which the NBV algorithm is not statistically different from the NF approach may be different for different environments.

From the means tabulated in Table 2, one can immediately identify a difference between the distance travelled by the NF approach and the NBV approach. This difference is also apparent in the time taken to finish the exploration and mapping process. Although such metrics have been used to compare different exploratory algorithms, it is important to note that in this case, the distance travelled and the time taken do not depend on just the adopted strategy. The actual dis-

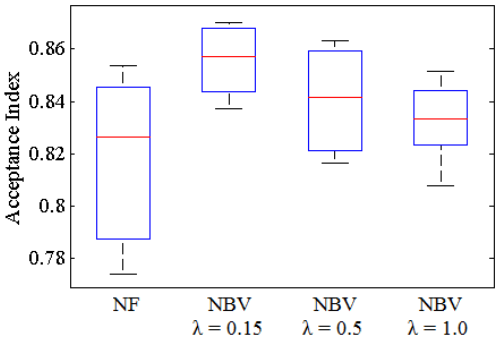


Figure 7: Boxplot of the acceptance index distributions.

Table 3: ANOVA statistical test: The Hypotheses.

H_0	The NF approach and the NBV approach with $\lambda = \{0.15, 0.5, 1.0\}$ all perform equally well.
H_1	Some of the approaches have better performance than the others in terms of map accuracy.

tance travelled by the robot does not only depend on the destination, but it also depends on the local trajectories planned by the local path planner, which accounts for the robot kinematic constraints. Hence, the length of the travelled path is typically longer than that which is considered as the global path, $L(q, q_k)$ in both algorithms. Furthermore, the time taken to finish the whole process does not only depend on the algorithm or the distance travelled. It also depends on the speed of the robot while it is navigating. This speed may be reduced significantly if the robot passes close to an obstacle, which is a precaution taken by the local path planner in order to avoid collision with obstacles. For these reasons, this study mainly focused on the accuracy of the map obtained by the two different algorithms which were implemented from scratch as ROS packages for the first time.

6 CONCLUSIONS

In this paper, a modular design concept was used in order to implement a robotic system that can autonomously explore, navigate and map an unknown environment in ROS. Furthermore, this work contributes a new package to the ROS community. This package consists of the implementation of two exploration algorithms which can be used independently of the navigation and the SLAM components. Moreover, the experimental evaluation of the *Nearest Frontier* (NF) and the *Next Best View* (NBV) approach revealed that in general, the NBV approach produces more accurate maps than the NF approach. Furthermore, from this study one can also conclude that as the parameter λ in the NBV approach is increased, the NBV algorithm converges to the NF approach. Moreover, the results clearly confirm that the best choice of an exploration strategy, is highly dependent on the problem at hand and the environment in question.

REFERENCES

Amigoni, F. (2008). Experimental evaluation of some exploration strategies for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA), 2008*, pages 2818–2823.

Burgard, W., Stachniss, C., Grisetti, G., Steder, B., Kummerle, R., Dornhege, C., Ruhnke, M., Kleiner, A., and Tardós, J. D. (2009). A comparison of SLAM algorithms based on a graph of relations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS)*., pages 2089–2095.

Carlone, L., Du, Jingjing, K., Ng, M., Bona, B., and Indri, M. (2014). Active SLAM and exploration with particle filters using Kullback-Leibler Divergence. *Journal of Intelligent and Robotic Systems*, 75(2):291–311.

Carpin, S. (2008). Fast and accurate map merging for multi-robot systems. *Autonomous Robots*, 25(3):305–316.

Dechter, R. and Pearl, J. (1985). Generalized best-first search strategies and the optimality of A*. *Journal of the ACM (JACM)*, 32(3):505–536.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Dissanayake, G., Huang, S., Wang, Z., and Ranasinghe, R. (2011). A review of recent developments in Simultaneous Localization and Mapping. In *6th International Conference on Industrial and Information Systems*, pages 477–482.

Fox, D., Burgard, W., Thrun, S., et al. (1997). The Dynamic Window Approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33.

Gonzalez-Baños, H. H. and Latombe, J.-C. (2002). Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11):829–848.

Grisetti, G., Stachniss, C., and W.Burgard (2007). Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46.

Makarenko, A. A., Williams, S. B., Bourgault, F., and Durrant-Whyte, H. F. (2002). An experiment in integrated exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, (IROS)*, pages 534–539.

Stachniss, C., Grisetti, G., and Burgard, W. (2005). Information gain-based exploration using rao-blackwellized particle filters. In *Robotics: Science and Systems*, volume 2, pages 65–72.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Proc. of IEEE International Conference on Robotics and Automation, 1994.*, pages 3310–3317.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*, chapter 10, pages 309 – 330. MIT press.

Thrun, S., Montemerlo, M., Koller, D., Wegbreit, B., Nieto, J., and Nebot, E. (2004). FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association. *Journal of Machine Learning Research*, 4(3):380–407.

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1997.*, pages 146–151.