Proceedings of Australasian Conference on Robotics and Automation, 2-4 Dec 2013, University of New South Wales, Sydney Australia

Nearest Neighbour Exploration with Backtracking for Robotic Exploration of Complex 3D Environments

Phillip Quin, Gavin Paul, Dikai Liu, Alen Alempijevic
University of Technology Sydney, Australia
phillip.d.quin@student.uts.edu.au
gavin.paul@uts.edu.au
dkliu@eng.uts.edu.au
alen.alempijevic@uts.edu.au

Abstract

This paper presents an extension to an exploration strategy called Nearest Neighbour (NN) Exploration to reduce required exploration time. The new approach, called Nearest Neighbours with Backtracking (NNB) involves keeping track of all neighbours at each time step throughout exploration in a tree structure. This strategy is shown through simulations to improve exploration time.

1 Introduction

There exist many environments in which it would be desirable to have robots perform exploration tasks rather than humans. The motivation for this could be financial, time-saving, or risk management. Examples of such environments are nuclear power plants [Nagatani *et al.*, 2011], bridge structures [Romey *et al.*, 2007], and inside ship hulls [Shang *et al.*, 2008]. Many of these environments involve complex and confined 3D spaces that the robot must navigate through whilst avoiding collisions.

Complex environments often preclude any option of manual guidance, as the robot configurations required to navigate such environments would make manual operation complex or inefficient. It is thus necessary for a robot to be able to perform its exploration task autonomously.

Exploration involves several key aspects. First, areas of interest must be identified. Second, the robot must determine how to move there, if possible. Third, the robot must be able to determine when all information worth collecting, or all the information that is physically possible to collect has been gathered, and terminate exploration.

The scenario in this paper involves a robot arm placed in a complex environment, at a fixed location. An RGB-D camera is affixed to the robot's end effector, and used to collect environment information. An octomap is used to store the location of known obstacles, free space, and unknown voxels [Wurm *et al.*, 2010].

This paper demonstrates that the Nearest Neighbour exploration approach [Quin *et al.*, 2013] can be made more efficient by maintaining a tree of the nearest neighbour poses. When reaching a pose that has no useful neighbours, previously it was necessary to use the AXBAM_NBV [Paul *et al.*, 2009; Quin *et al.*, 2013] algorithm to perform a large sampling of configuration space to find the next starting pose for NN exploration. The proposed alternative is to instead backtrack through the tree of nearest neighbours and select a pose that was not selected earlier. If this backtracking step does not yield a worthwhile pose then a search through the larger configuration space sample is necessary.

The rest of this paper is structured as follows. Section 2 presents an overview of existing approaches to robot exploration. Section 3 presents the Nearest Neighbour Exploration with Bactracking (NNB) approach. Section 4 details the results of simulation experiments. Finally, in Section 5, conclusions are drawn from the results, and future work is suggested.

2 Related Work

Autonomous exploration is related to the Watchman Problem, in which an agent must find and follow a path which allows it to cover with its field of view an entire area or volume as quickly as possible. A similar problem is the Art Gallery Problem, which involves placing the fewest number of stationary guards or cameras, to completely cover the area or volume of the gallery. In the case of a single robot, using the solution to the Art Gallery Problem involves solving the Travelling Salespan Problem, where the different goal positions are the optimal positions of the stationary guards. These problems, which involve environments that are completely known apriori, are NP-hard [Krause, 2007; Hawley, 2009]. Most exploration strategies for environments that are not completely known apriori use greedy approaches to determine where the robot should move next to make a new observation. This is often called determining the Next Best View (NBV) [Connolly, 1985;

Pito, 1996].

Many solutions to the NBV problem exist. Frontier-based exploration detects boundaries between free space and unknown space, and moves the robot to frontiers until no more reachable frontiers exist [Yamauchi *et al.*, 1998]. Variants of this exploration strategy have been used with eye-in-hand robot manipulators [Dornhege and Kleiner, 2011]. Another family of solutions to the NBV problem involves potential fields, in which obstacles repel the robot and the goal attracts the robot, so that the path the robot must take is given by the gradient of steepest descent from the robot's initial position to some local minima [Khatib, 1985; Kim and Khosla, 1992]. This approach is used for exploration by setting all frontiers to be goal positions [Shade and Newman, 2011].

There exist exploration strategies that utilise configuration space (C-space) [Torabi *et al.*, 2007; Yu and Gupta, 2004; Huang and Gupta, 2005]. The goal is to explore the environment the robot is located in, and additionally to explore space in the environment that will potentially increase the robot's possible range of motion, thus allowing it to more effectively explore the environment.

Another way in which C-space can be used to explore is as the space of possible viewpoints from which the NBV will be chosen. This approach is particularly used when the robot is a complex sequence of links that may be highly redundant [Paul *et al.*, 2009]. In this case, finding NBVs in Euclidean space would potentially require computationally expensive inverse kinematics to determine the required robot configuration. Selecting NBVs from C-space means such calculations can be avoided.

In [Paul *et al.*, 2009; 2011], with an exploration strategy named AXBAM, NBVs are selected from a large sample set of poses drawn from C-space. Forward kinematics are used to compute a sensor position given a robot pose, and raytracing is used to estimate the expected information gain that would result from a sensor reading at that position. The viewpoint with the highest information gain weighted against required robot effort, is selected as the NBV.

The authors' previous work in [Quin *et al.*, 2013] demonstrated a Nearest Neighbour (NN) approach in which the NBV is chosen from smaller sets sampled from C-space surrounding the robot's current position. When such a small set did not result in any valid NBV, the entire sample C-space set was reevaluated.

Evaluating each sample of the entire configuration space is costly, and the advantage of NN over AXBAM lies in delaying such a costly evaluation of the sample set from all C-space. This paper proposes and demonstrates the effectiveness of a new variant of NN, called NNB, which increases this advantage further.

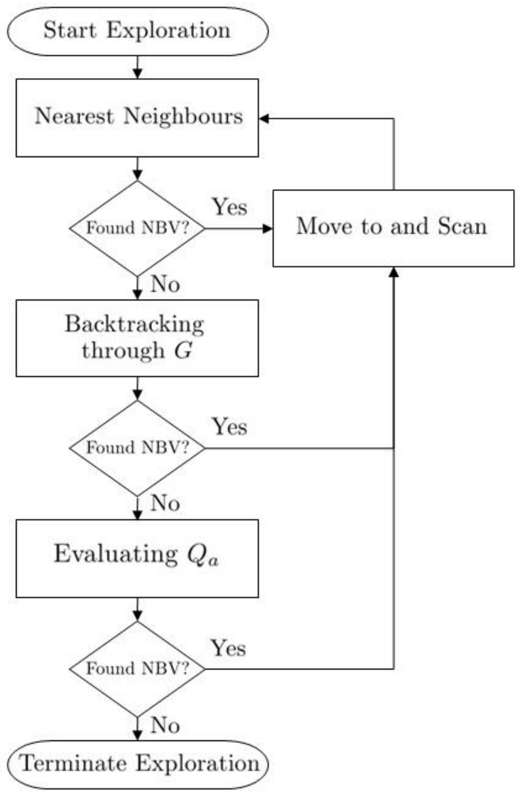

Figure 1: A flowchart of the components of Nearest Neighbours Exploration with Backtracking.

3 Nearest Neighbours Exploration with Backtracking

This section describes the Nearest Neighbour with Backtracking (NNB) algorithm (see Figure 1). There are three methods for determining the NBV; methods are only used when the previous method has failed. In order, they are: evaluating nearest neighbours, backtracking through the tree of nearest neighbours, and finally, evaluating all poses in a sampling of C-space. If all methods fail, exploration is terminated.

3.1 Initialising Exploration

In NNB exploration, a robot with j degrees of freedom (jDOF) begins in some pose q_{curr} which is a vector of joint angles $(\theta_1, \theta_2, ...\theta_j)$. A tree, G is initialised to have the pose q_{curr} as its root, and a pointer, c is initialised to point at the root, marking the robot's current position in G. A map M, which is the robot system's internal representation of the environment, is set to have a certain amount of space known to be initially free so as to allow the robot to begin exploration. In this paper, M is a map which divides space into equally sized voxels representing either unknown space, free space, or occupied space.

A set of robot poses, Q_a is generated, which contains

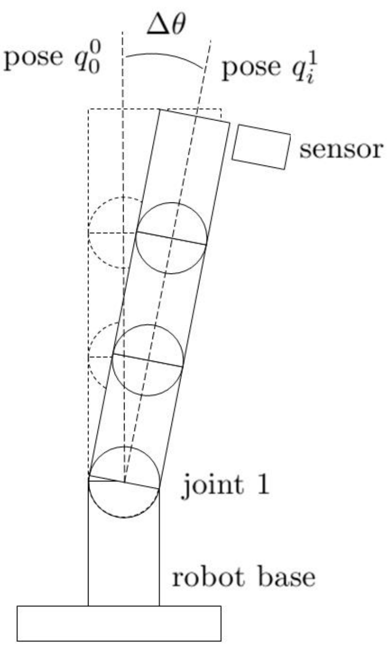

Figure 2: Defining $\Delta\theta$ for a 3DOF robot so that knowing pose q_0^0 and its neighbour q_i^1 are collision-free means there cannot be obstacles in the path between them.

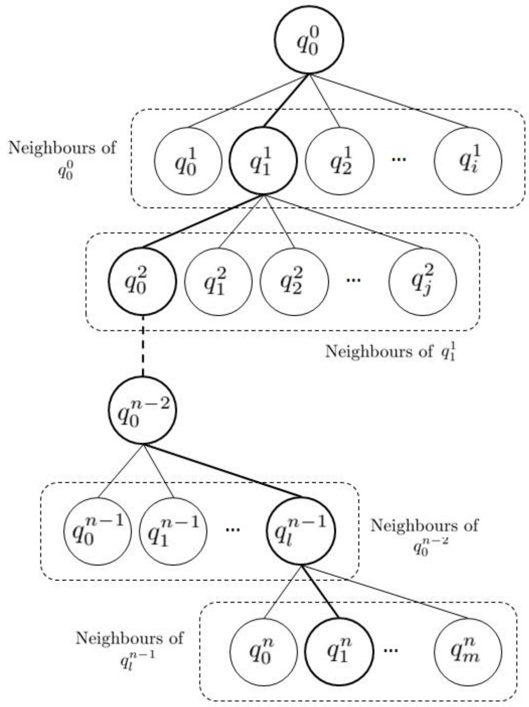

Figure 3: The tree G of robot poses in C-space formed in nearest neighour exploration.

sample poses in C-space. When no pose in Q_a is determined to have a high enough value of information (above a threshold), exploration will terminate. In this way, the size of Q_a is proportional to the minimum amount of information that will be collected by the exploration algorithm.

3.2 Nearest Neighbours

A set of poses, Q_n, in the neighbourhood of the current pose is generated. In this paper, Q_n is populated by iterating over each joint angle θ_i in q_{curr}, where $i \in [1...j]$. Added to Q_n are poses that are equal to q_{curr} but in which θ_i is increased by a small angle increment $\Delta\theta$ and a pose equal to q_{curr} but in which θ_i is decreased by $\Delta\theta$. This results in $2 * j$ poses being added to Q_n.

It is possible to choose $\Delta\theta$ in such a way that path planning between q_{curr} and any pose in Q_n is unnecessary (see Figure 2). Knowing that each pose will not result in a collision is sufficient to guarantee no obstacles exist between the poses; it is therefore possible to move directly from one pose to the other. Any poses in Q_n which would result in a collision with either known obstacles, or unknown space, are removed.

Next, poses in Q_n must be evaluated to determine which will lead to the greatest amount of new information. Given a pose q, forward kinematics are used to calculate the homogenous transform matrix ${}^oT_s^q$ which represents the sensor's position and orientation in Euclidean space relative to the robot base. The information gain of each pose $q \in Q_n$ is then estimated by ray-tracing from ${}^oT_s^q$ into M to the ends of rays describing the sensor's range and field of view (FOV). For each ray traced, information gain is the number of unknown voxels counted before reaching either the end of the ray, or an occupied voxel.

The pose $q \in Q_n$ with the highest information gain value, provided it is above a chosen information threshold τ_n (representing the minimum amount of expected information a viewpoint must have to be considered as a potential NBV), is selected as q_{nbv}. Poses with information values below τ_n are removed from Q_n. The poses in Q_n are then added to G as the children of q_{curr}.

The robot moves to q_{nbv} and collects a sensor reading which is incorporated into M. The pointer, c is changed to indicate the appropriate node in G.

The neighbour generation and evaluation steps, followed by motion and sensor scans, are repeated until such time as Q_n is empty after pose evaluation.

3.3 Backtracking through G

At the beginning of the backtracking step, the tree G has a depth of n. Each level of the tree has x number of nodes, where:

$$1 <= x <= 2 * j \tag{1}$$

On each level other than the n^{th}, one node is parent to all the nodes of the next level. c points at the leaf node in G that indicates the robot's current position. G and c, along with the threshold value τ_n, are given to the Nearest Neighbour Bactracking algorithm (Algorithm 1).

Algorithm 1: NearestNeighbourBacktracking(G, c, τ_n)

```
NBV ← $\varnothing$
while not isRoot$(c, G)$
  do
    $n \leftarrow c$
    $c \leftarrow$ parent$(c, G)$
    removeNode$(n, G)$
    for each $node \in$ children$(c, G)$
      do
        if information$(node) < \tau_n$
          then removeNode$(node, G)$
          else if information$(node)$
        > information(NBV)
          then NBV ← $node$
        if NBV $\neq \varnothing$
          then break
output (NBV)
```

In Figure 3, an example tree G is shown, with the path from the root q_0^0 to the current position q_1^n shown with bold edges.

The NBV is found by repeating the following steps (shown in brackets are the results of the first iteration on the graph G in Figure 3):

1. If the current node is the root, then terminate backtracking.
2. Change c to point to the parent of the current node. This is the new current node (i.e. c now points at q_1^{n-1}).
3. Remove the node previously pointed to by c (remove q_1^n).
4. Evaluate the children of the current node. Remove any with information gain below τ_n
5. If children remain, select the child with the best information gain to be q_{nbv} and terminate backtracking. Else repeat from step 1.

If a q_{nbv} has been selected, the robot moves from its current position to this goal position. The path taken is given by the sequence of nodes visited when backtracking.

Otherwise, G is empty, and q_{nbv} must determined by evaluating the poses in Q_a.

3.4 Evaluating Q_a

The last step of the NNB exploration algorithm involves evaluating poses in Q_a to find a NBV. First, each pose $q \in Q_a$ that results in a potential collision with known occupied voxels in M is removed from Q_a. Remaining poses are added into set Q_{unk} if they would place the robot in collision with any unknown voxels in M, and otherwise into set of possible poses Q_{poss} if there is no such collision.

Each pose $q \in Q_{poss}$ is evaluated to determine its expected value in information gain, denoted by i_q. Poses with information gain below a threshold τ_a are removed from Q_{poss}, τ_a may be the same value as τ_n.

The remaining poses $q \in Q_{poss}$ are also given a value $e_q = e([q_{curr}, q])$, denoting the joint effort required to move to that position from q_{curr} (2).

$$e([q_1, ..., q_n]) = \sum_{i=2}^{n} \|q_i - q_{i-1}\| \qquad (2)$$

After ranking all poses q by the ratio of i_q to e_q, the best q is chosen as the NBV. This NBV becomes the new root of G.

If Q_{poss} is empty after checking for collisions and evaluating information gain, then NNB exploration terminates.

4 Experiments and Results

The NNB algorithm was evaluated through a series of simulations run on an Intel Core 2 Duo (3.00 GHz) with 3.5 GB of RAM in MATLAB. Mex files were used to interface between MATLAB code and C++ Octomap libraries (v. 1.5) [Wurm *et al.*, 2010]. The robot model used is a 7DOF robot based on work by [Pagano *et al.*, 2012].

As shown in Figure 4, six simulated environments were used. The most open environment was a floor environment, with no obstacles other than the plane the robot was attached to. The most confined environment was a tunnel offering little maneuvering space, and in which much of the environment near the robot would be inside the sensor minimum range for most possible robot configurations. Other environments spanned between the two extremes.

Various metrics for comparison of algorithms were collected as part of each experiment, these were:

- **Time**: minutes the algorithms took to run to completion,
- **Information**: voxels known to be free or occupied as a result of exploration,
- **Joint Effort**: given the robot's trajectory $Q_{traj} = [q_1...q_n]$, where each q is a vector of joint angles, the joint effort is given by $e(Q_{traj})$ (2).
- **Parallel Joint Effort**: allowing for the fact that joints can move in parallel, this measures the most effort required over the robot's trajectory, which

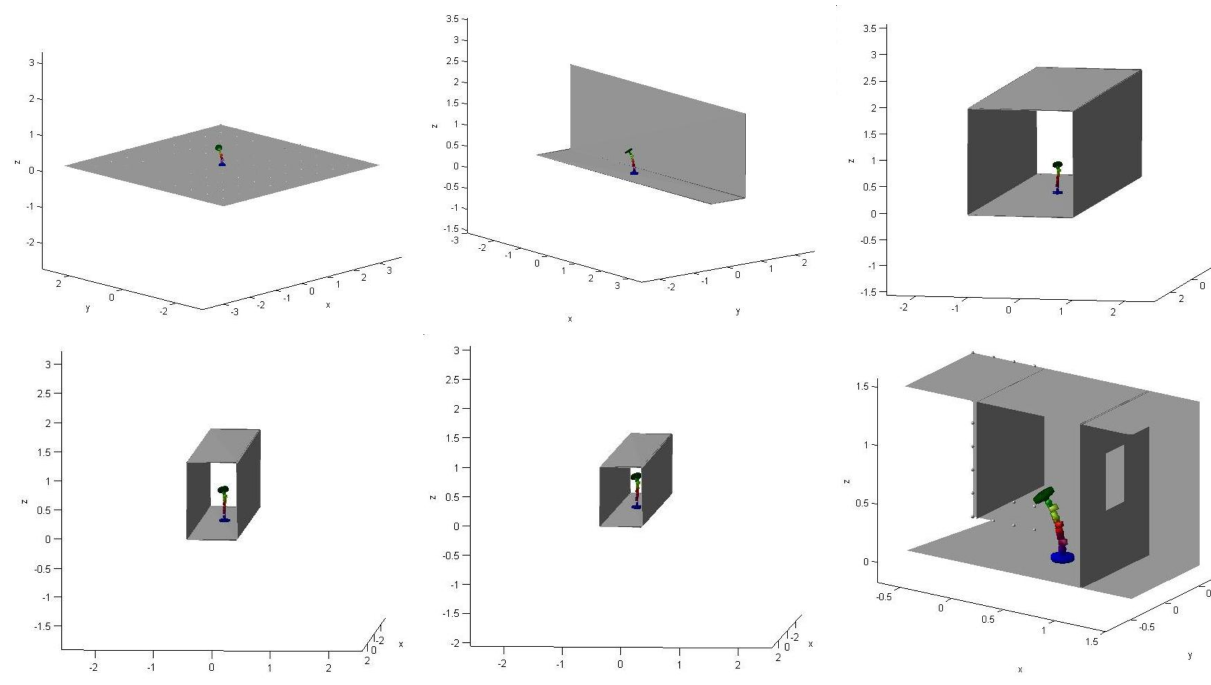

Figure 4: Simulated environments: starting from top left moving clockwise are the floor, the half tunnel, big tunnel, plate, confined tunnel, and tunnel environments. Dimensions shown in meters.

gives an intuition as to how long it might take the robot to move through the sequence of poses. Given Q_{traj}, this value is given by $e_{par}(Q_{traj})$ (3).

$$e_{par}([q_1, ..., q_n]) = \sum_{i=2}^{n} \max(|q_i - q_{i-1}|) \qquad (3)$$

In order to provide comparison for the performance of NNB, simulations were also run with two other algorithms. These were NN, which resembles NNB but has no backtracking step, and AXBAM, which has no nearest neighbours or backtracking step and evaluates Q_a at each time step.

Exploration was performed giving each algorithm a set Q_a containing 3^7 poses, then again with Q_a containing 4^7 poses, and finally, with Q_a containing 5^7 poses in order to determine the effect of increased $|Q_a|$ on exploration time, and information collected.

Finally, each exploration simulation was performed three times so that differences in performance due to the non-deterministic nature of the RRT robot path planner could be accounted for.

Figure 3.4 illustrates the information collected by each run of the exploration algorithms in each environment. For each of the six environments, the highest amount of information collected by any of the twenty seven runs (three algorithms, three sizes of Q_a, run three times each) was selected as the maximum value of information for that environment. Information collected by each run is shown as a proportion of this maximum.

Figure 5 shows how long each algorithm took to run in each environment. AXBAM performs poorest in each case. For most of the environments NNB and NN perform similarly in terms of information collecte. NN and NNB both collect more information than AXBAM in a shorter period of time. As expected when $|Q_a|$ is larger, each algorithm takes longer to terminate, though this difference is particularly pronounced for AXBAM. NNB is faster than simple NN exploration in almost all cases. This difference is small when $|Q_a|$ is small, 5% to 37% depending on the environment. When $|Q_a|$ is larger, with 5^7 poses, this trend continues and the difference is more pronounced; NNB is 26% to 55% faster than NN depending on the environment. In all cases, NNB and NN terminate sooner than AXBAM with more information collected.

As seen in Figure 3.4, a disadvantage of NNB is that the total Joint Effort tends to be more than for NN, this remains true for total Parallel Joint Effort. The reason for this can be seen in an example in Figure 8. Given the robot's current position at q_1^n, if the NBV selected is the node q_j^2, then the robot's path from current position to the goal would be given by the path, P_G of length, l in G leading from q_1^n to q_j^2. It is possible that a path planner given q_1^n and q_j^2 as start and goal positions, and M defining free space and obstacles, would find a more

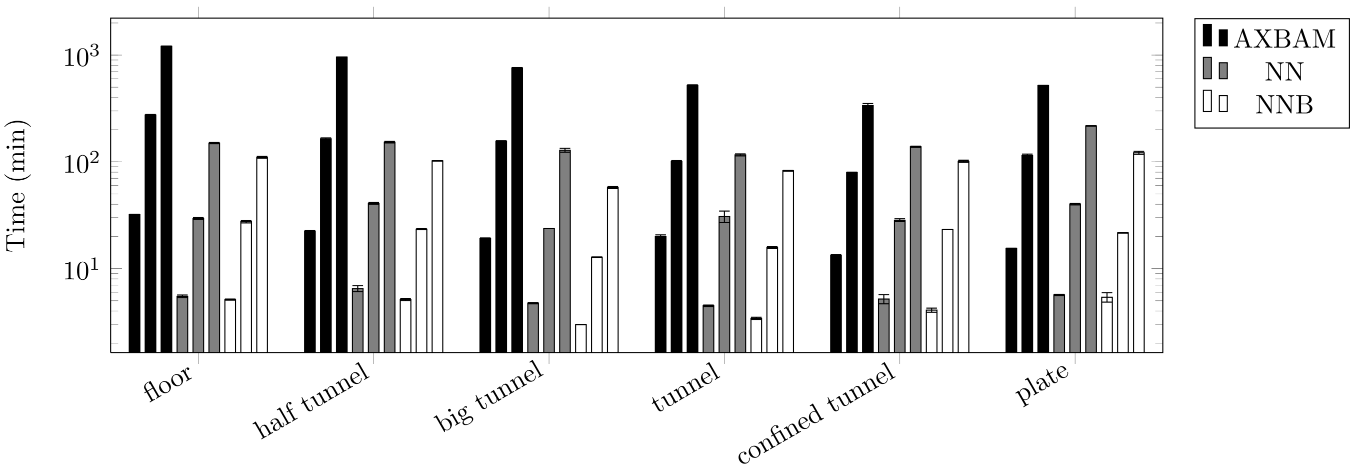

Figure 5: Time taken by each algorithm in each environment, at $|Q_a| = 3^7$, $|Q_a| = 4^7$, and $|Q_a| = 5^7$

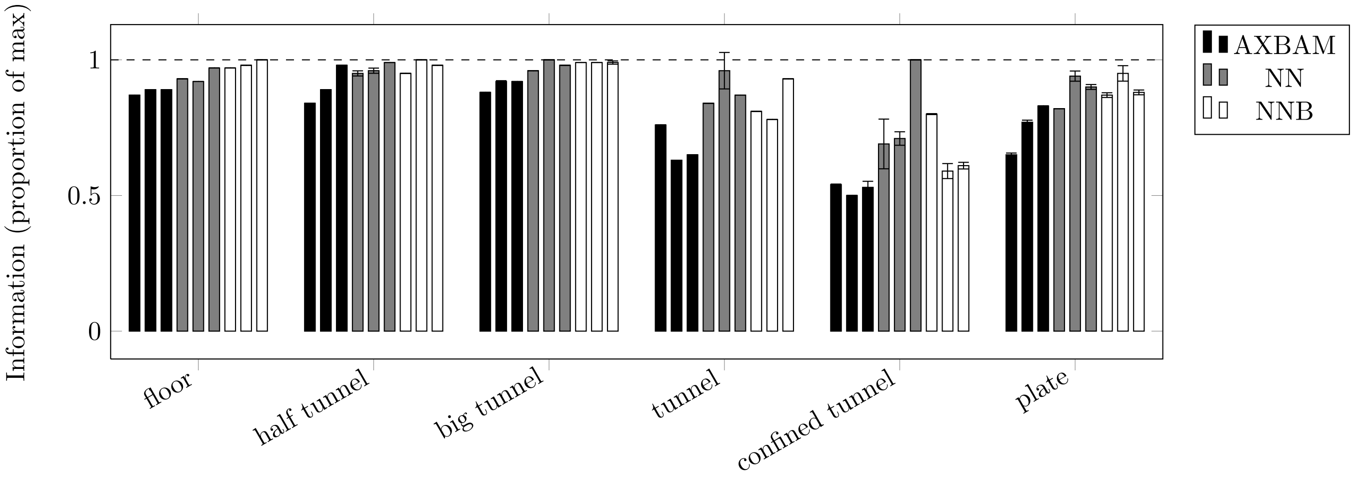

Figure 6: Information collected by each algorithm in each environment, at $|Q_a| = 3^7$, $|Q_a| = 4^7$, and $|Q_a| = 5^7$

direct path $P_{planner}$ with length $l_{planner} < l$.

The time taken by a robot to complete exploration of an environment is a combination of the time required to compute the NBV, the time taken to scan (assumed to be 0.03s in these simulations), and the time for the robot to physically move to each NBV. The choice of which algorithm out of the three compared in this paper depends on the relative times taken for each of these tasks. AXBAM, which requires the fewest scans to explore an environment is best suited when scanning is a lengthy process. NN, which results in low parallel effort required for exploration, is best suited for slow moving robots. NNB is best suited when scanning and movement are relatively quick and the time to compute the NBVs is the factor that needs to be minimised (see Figure 9).

5 Conclusions and Remarks

This paper has described Nearest Neighbours with Backtracking (NNB), which is an extension of Nearest Neighbours exploration. This approach has been implemented and evaluated in simulations and shown to result in faster exploration. However, NNB results in higher total joint effort, and therefore in more robot motion. Thus, this paper recommends suitable exploration strategies for different sensor acquisition and manipulator joint velocity combinations.

Further experiments will be performed to determine the effect of thresholds τ_n and τ_a, and how these might best be selected to increase efficiency of exploration.

Field trials of exploration using the NNB algorithm will be performed to verify the work done in simulations.

Acknowledgments

This work is supported by the Australian Research Council (ARC) Linkage Grant (LP100200750), the NSW Roads and Maritime Services, and the Centre for Autonomous Systems (CAS) at the University of Technology, Sydney.

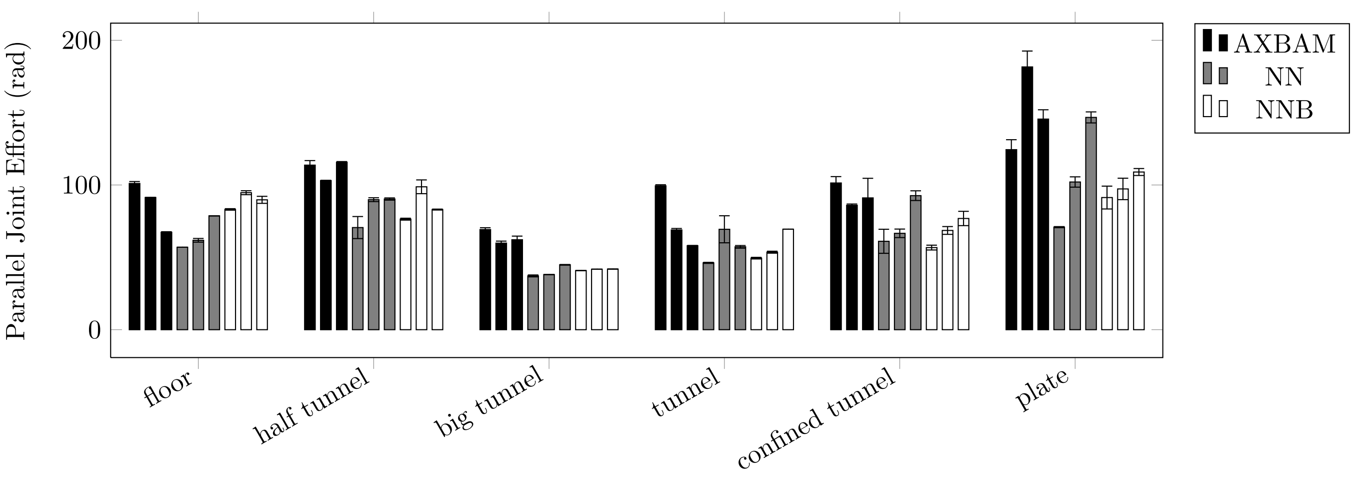

Figure 7: Parallel effort required by each algorithm in each environment, at $|Q_a| = 3^7$, $|Q_a| = 4^7$, and $|Q_a| = 5^7$

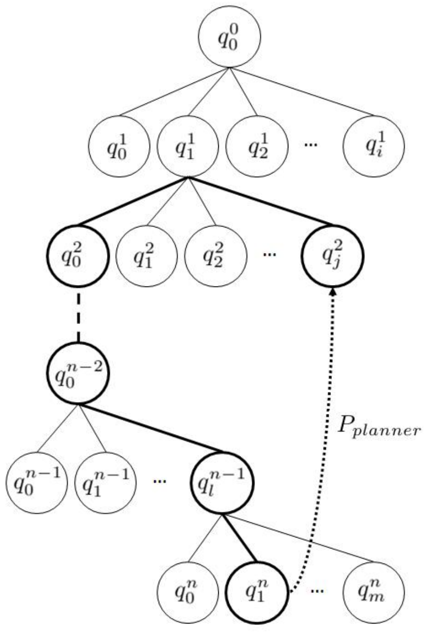

Figure 8: P_G from q_1^n to q_j^2 shown in bolded edges, and $P_{planner}$ for the same start and goal nodes shown by the dashed arrow.

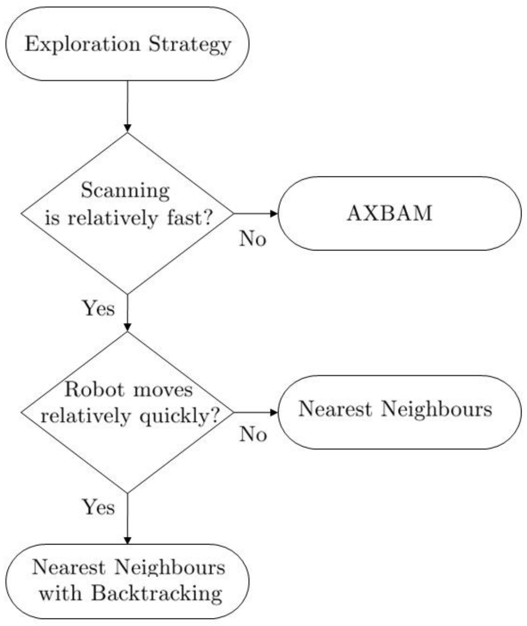
Exploration Strategy
Scanning is relatively fast?
No
AXBAM
Yes
Robot moves relatively quickly?
No
Nearest Neighbours
Yes
Nearest Neighbours with Backtracking

Figure 9: Choosing which algorithm based on time taken for scanning, moving, and determining NBV.

References

[Connolly, 1985] C. Connolly. The determination of next best views. In *Robotics and Automation. Proc. IEEE International Conference on*, volume 2, pages 432–435, 1985.

[Dornhege and Kleiner, 2011] C. Dornhege and A. Kleiner. A frontier-void-based approach for autonomous exploration in 3D. In *Safety, Security, and Rescue Robotics (SSRR), IEEE International Symposium on*, pages 351–356, 2011.

[Hawley, 2009] John Hawley. Hierarchical task allocation in robotic exploration. Master's thesis, Rochester Institute of Technology, 2009.

[Huang and Gupta, 2005] Yifeng Huang and K. Gupta. An adaptive configuration-space and work-space based criterion for view planning. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 3366–3371, 2005.

[Khatib, 1985] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation, Proc. IEEE International Conference on*, volume 2, pages 500–505, 1985.

[Kim and Khosla, 1992] J.-O. Kim and P. K. Khosla. Real-time obstacle avoidance using harmonic potential functions. 8(3):338–349, 1992.

[Krause, 2007] Andreas Krause. Near-optimal observation selection using submodular functions. In *AAAI Nectar*, 2007.

[Nagatani *et al.*, 2011] Keiji Nagatani, S. Kiribayashi, Y. Okada, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, and Y. Hada. Redesign of rescue mobile robot quince. In *Safety, Security, and Rescue Robotics (SSRR), IEEE International Symposium on*, pages 13–18, 2011.

[Pagano *et al.*, 2012] D. Pagano, Dikai Liu, and K. Waldron. A method for optimal design of an inchworm climbing robot. In *Robotics and Biomimetics (ROBIO), IEEE International Conference on*, pages 1293–1298, 2012.

[Paul *et al.*, 2009] Gavin Paul, Dikai Liu, Nathan Kirchner, and Gamini Dissanayake. An effective exploration approach to simultaneous mapping and surface materialtype identification of complex three-dimensional environments. *J. Field Robot.*, 26:915–933, 2009.

[Paul *et al.*, 2011] Gavin Paul, Stephen Webb, Dikai K. Liu, and Gamini Dissanayake. Autonomous robot manipulator-based exploration and mapping system for bridge maintenance. *Robotics and Autonomous Systems*, 59(78):543–554, 2011.

[Pito, 1996] R. Pito. A sensor-based solution to the "next best view" problem. In *Pattern Recognition, Proceedings of the 13th International Conference on*, volume 1, pages 941–945, 1996.

[Quin *et al.*, 2013] Phillip Quin, Gavin Paul, Alen Alempijevic, Dikai K. Liu, and Gamini Dissanayake. Efficient neighbourhood-based information gain approach for exploration of complex 3d environments. In *Robotics and Automation (ICRA), Proc. IEEE International Conference on*, pages 1335–1340, 2013.

[Romey *et al.*, 2007] P. Romey, A. Nhan, K. Williams, and M. Dunn. Sydney harbour bridge conservation management plan 2007. Technical report, Roads and Traffic Authority, 2007.

[Shade and Newman, 2011] Robbie Shade and Paul Newman. Choosing where to go: Complete 3D exploration with stereo. In *Robotics and Automation (ICRA), Proc. IEEE International Conference on*, pages 2806–2811, 2011.

[Shang *et al.*, 2008] J. Shang, S. Mondai, A. A. Brenner, B. Bridge, and T. Sattar. A cooperative climbing robot for melt weld inspection on large structures. *Advances in Mobile Robotics*, pages 47–54, 2008.

[Torabi *et al.*, 2007] L. Torabi, M. Kazemi, and K. Gupta. Configuration space based efficient view planning and exploration with occupancy grids. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2827–2832, 2007.

[Wurm *et al.*, 2010] Kai M. Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. of the ICRA 2010 workshop*, 2010.

[Yamauchi *et al.*, 1998] B. Yamauchi, A. Schultz, and W. Adams. Mobile robot exploration and map-building with continuous localization. In *Robotics and Automation, Proc. IEEE International Conference on*, volume 4, pages 3715–3720, 1998.

[Yu and Gupta, 2004] Yong Yu and Kamal K. Gupta. C-space entropy: A measure for view planning and exploration for general robot-sensor systems in unknown environments. *I. J. Robotic Res.*, 23(12):1197–1223, 2004.