Efficient Neighbourhood-Based Information Gain Approach for Exploration of Complex 3D Environments*

Phillip Quin, Gavin Paul, Alen Alempijevic, Dikai Liu and Gamini Dissanayake
Center for Autonomous Systems
University of Technology Sydney, Sydney 2007, Australia
phillip.d.quin@student.uts.edu.au, gavin.paul@uts.edu.au, alen.alempijevic@uts.edu.au
dkliu@eng.uts.edu.au, gdissa@eng.uts.edu.au

***Abstract*— This paper presents an approach for exploring a complex 3D environment with a sensor mounted on the end effector of a robot manipulator. In contrast to many current approaches which plan as far ahead as possible using as much environment information as is available, our approach considers only a small set of poses (vector of joint angles) neighbouring the robot's current pose in configuration space. Our approach is compared to an existing exploration strategy for a similar robot. Our results demonstrate a significant decrease in the number of information gain estimation calculations that need to be performed, while still gathering an equivalent or increased amount of information about the environment.**

I. INTRODUCTION

Many environments which require detailed inspection are hazardous for humans or too time consuming to maneuver through and explore. Some examples are mines, tunnels, and bridges. Using robots to explore these environments will reduce potential hazards to workers and also allow for exploration of confined spaces that human workers might have difficulty reaching.

Ideally, a robot should be able to explore an environment autonomously, without guidance from an operator. This would enable exploration of an environment even when communication with the robot is not feasible. Exploring autonomously requires the robot to intrinsically determine (a) safety - where it is safe to move, (b) efficiency - the shortest sequence of movements which will gather descriptive information about the environment, and (c) completeness of information - if there is any information left to discover.

Several existing strategies [1], [2], [3] select the goal position and associated path for the end effector based on maximising expected information gain whilst utilizing the existing environment representation. While these approaches were suitable for sensors which have a large acquisition time (such as a tilting laser rangefinder), the advancements of 3D sensors such as the Microsoft Kinect allow us to simultaneously capture information as the end effector moves through the environment. Hence, it is possible to update the environment map not only at the goal position, but also along the calculated path. It is probable that information gathered whilst in motion would suggest several new, more optimal, paths than the one currently being followed. Consequently, one disadvantage of current approaches is that information discovered along a trajectory will not be taken into account until the trajectory is completed.

The approach presented in this paper builds upon our previous next-best-view exploration algorithm used as part of the Autonomous eXploration to Build a Map system [3]. The AXBAM Next Best View algorithm (AXBAM_NBV) is based on sampling the robot's configuration space (c-space) and determining the next pose based on estimated information gain and the effort required. While many current approaches to exploration plan as far ahead as possible using all of the available map information, our approach instead attempts to reduce the number of gain calculations required. This is done by considering only the poses closest to the robot's current position and delaying evaluation of poses further afield.

Our exploration scenario involves a robot arm placed at a fixed location within the environment. The sensor used to perceive the environment is an RGB-D camera attached to the end effector. Our goal is to gather as much information about the environment around the robot as possible and store the location of free space and obstacles using an octomap [4].

This paper demonstrates the results of a new approach called Nearest Neighbour Next Best View (NN_NBV), where the robot first considers nearby poses in c-space until it reaches a dead-end (either in terms of possible motion or information gain), in which case it selects a pose from a larger set distributed evenly amongst its c-space. These results are then compared to results when exploring with AXBAM_NBV.

NN_NBV is shown to result in significantly fewer information gain calculations being performed, therefore reducing computation time. It also results in either comparable or reduced robot movement, meaning the total exploration time including computation is also reduced, while still gathering comparable or increased information about the environment.

The remainder of this document is organized as follows: Section II presents an overview of existing approaches to exploration. Section III details the Nearest Neighbour Next Best View approach and results are provided in Section IV. Finally, conclusions are drawn and future work proposed in Section V.

*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.

II. RELATED WORK

Though autonomous exploration has been an active field of research for some time, finding a single optimal path to explore an entire environment has been found to be NP-hard [5], [6]. For this reason, current exploration strategies tend to approach exploration in a greedy fashion.

Frontier-based exploration strategies such as [7] examine a frontier and then move towards the closest or largest frontier until no more frontiers are reachable. Frontier-based exploration has been combined with potential field exploration and shown to be an effective method of exploration [8]. However, while Shade and Newman's approach is able to run in real-time for 2D environments, it is computationally expensive when used in 3D environments.

Exploration strategies can either examine c-space, the workspace; or typically both. C-space allows for path planning for a robot manipulator with a fixed base and does not require inverse kinematics to be calculated. C-space exploration strategies become intractable as the c-space dimensionality increases. There has been work in analyzing the growth of c-space and determining how to explore it efficiently [9]. Typically this means sampling points in c-space rather than fully creating the space. Rapidly-exploring Random Trees (RRTs) and various other path planning strategies are examples of such c-space sampling approaches [10]. Work has been done in combining the benefits of both workspace and c-space exploration by switching between the two as needed [11].

Potthast and Sukhatme [1] give an example of frontier-based exploration using probabilistic road maps and an information gain metric that counts unknown voxels. Their approach is demonstrated using a PR2 robot equipped with a Kinect to explore a table covered in objects. This approach selects a goal pose from a set of possible goal poses based on information gain. A set of paths to get to this selected goal position is then considered based on probabilistic road maps. An optimal path is chosen by selecting the one which gathers the most information along its entire trajectory. However, all calculations are performed on a current map and the robot's motion is not adapted mid-trajectory to take into account information acquired whilst in motion. Hence, during the time required to perform these calculations several less optimal paths could have been chosen and completed, resulting in more captured information than would have been gathered with the single more optimal path. It is also possible that information gathered partway through a trajectory reveals several more promising directions in which information can be gathered, and in Potthast and Sukhatme's approach, would not be taken advantage of.

Instead of starting with a sample set of poses in c-space, it is possible to work backwards from the workspace and generate candidate members from c-space [2]. Dornhege and Kleiner do this by detecting frontier cells and voids (collections of unknown cells), generating possible viewpoints, determining which are valid in c-space and then choosing the one with the most gain. In order for the algorithm to terminate, an initially bounded search space is assumed. Since our robot must explore a space with unknown boundaries and obstacles that may seriously limit its mobility, our approach cannot use this assumption.

In the above approaches with 5 or 6DOF robots and even the PR2 robot manipulator (7DOF), substantial calculation is performed evaluating viewpoints which are never used. An optimal method for reducing information gain calculations would only update the gain information of viewpoints known to have changed after the last scan was taken. This can become difficult to determine though, particularly when the robot gathers information as it moves from one viewpoint to another, meaning that in the worst case all possible remaining poses must have their estimated gain recalculated. Our approach reduces the number of gain calculations required as much as possible. Thus, only poses closest to the robot's current position are considered and evaluation of future poses is delayed.

III. NEAREST-NEIGHBOURS NEXT BEST VIEW ALGORITHM

In this section an outline of the algorithm will be given, followed by an analysis of the estimated information gain calculations required.

A. The Algorithm

When observing a scene, the human eye makes fast motions called saccades [12]. These motions orient the focal area of the eye (the fovea) to center on the object or position of interest. When searching for a target, saccades do not randomly move the eye's focus around the scene, but can use the information that has been seen so far to determine the most likely position of the target [13].

It is therefore intuitive for a robot to use what it has seen so far to select a neighbouring viewing position (analogous to a person's peripheral vision) that will give the most information, and to reorient itself to observe the scene from that position.

In the NN_NBV exploration algorithm (Algorithm 1), neighbour poses, Q_n, need to be generated (Algorithm 2), and the best chosen from among them. Beginning with the current pose $q_{curr} = (\theta_1, ..., \theta_j)$, a vector of j joint angles, each joint angle in q_{curr} is iterated over, adding or subtracting a chosen angle Δ_q. The resulting pairs of poses are then added to Q_n.

This results in c neighbouring poses, where $c = 2*j$, since two neighbours are created for each joint. The step size Δ_q is chosen in such a way that any neighbouring poses are close enough to the current pose that checking for obstacles before moving to them is unnecessary; it is possible to assume that if both the current and the neighbour pose are valid then it is safe to move directly from one to the other.

A function to check pose validity is used, $V()$, which takes in a set Q of poses, and an octomap M. Each pose in Q is a vector of length j where j is the number of joints, e.g. $(\theta_1, ..., \theta_j)$. Function $V()$ returns subsets of poses under consideration Q: Q_p, containing valid poses assuming

unknown space is occupied, and Q_u, poses that are only valid if the unknown space in the map is assumed to be free space. Poses that cause collisions with occupied voxels or the robot itself are discarded.

The function $G()$, which takes in a pose and the octomap, raytraces into the octomap and returns the number of unknown voxels encountered.

Note that it is unnecessary to calculate or compare the effort of moving from q_{curr} to any pose $q \in Q_n$; the effort for all poses is the same by construction. Since both poses are close enough that path planning to avoid obstacles is not necessary, the effort to move between poses is equivalent to the actual joint motion required by the robot to move to the new position

These steps are then repeated (e.g. generate neighbour poses, select a pose, move to the pose, and scan from that pose) until none of the neighbouring poses has a gain value greater than some threshold, or when no neighbour poses are valid. This may or may not mean that the entire environment has been explored. To be sure that the environment has been adequately explored, the AXBAM_NBV strategy which samples the entire c-space, is invoked. This means checking the set of sample poses from c-space and finding the one with the highest non-zero gain/effort ratio, or terminating if there are no valid poses with greater information gain than a predetermined threshold.

Algorithm 1: $NN\_Next\_Best\_View$

Input: q_{curr} ⟵Current robot pose , M ⟵Octomap, Q ⟵Set of poses under consideration
Output: q_{nbv}, Q

$Q_n = Generate\_Neighbour\_Poses(q_{curr}, M)$;
$q_{nbv} = \emptyset$;
$best\_neighbour\_gain = 0$;
if $Q_n \neq \emptyset$ **then**
  for $q \in Q_n$ **do**
    $gain = G(q, M)$;
    if $gain > best\_neighbour\_gain$ **then**
      $best\_neighbour\_gain = gain$;
      $q_{nbv} = q$;
if $q_{nbv} == \emptyset$ **then**
  $q_{nbv}, Q = AXBAM\_NBV(q_{curr}, M, Q)$;

B. Complexity of Calculations

The number of gain calculations required depends on how many optimally chosen scans are required to cover the environment. Each scan taken requires c gain calculations, one for each neighbour (or $c-1$ if the robot's past trajectory is remembered and repeated calculations avoided).

If for each pose chosen by AXBAM_NBV (Algorithm 3), α nearest neighbour poses have to be chosen ($\alpha > 0$) to gather approximately the same amount of information, and assuming c is the constant number of nearest neighbours considered at each step, then the best case running time is:

Algorithm 2: $Generate\_Neighbour\_Poses$

Input: $q_{curr} = (\theta_1, ..., \theta_j)$, M ⟵Octomap, Δ_q ⟵Step size
Output: Q_n

for $i \in [1, j]$ **do**
  $q_{new} = q_{curr}$;
  $q_{new}.\theta_i = q.\theta_i + \Delta_q$;
  $Q_n = Q_n \cup q_{new}$;
  $q_{new}.\theta_i = q.\theta_i - \Delta_q$;
  $Q_n = Q_n \cup q_{new}$;
$Q_n, Q_u = V(Q_n, M)$;

$$N_{C_{min}} = e \times \alpha \times c \quad (1)$$

The worst case results in the same gain calculations as required for AXBAM_NBV, plus c gain calculations each time the robot moves to a new viewpoint and determines that no neighour poses result in any worthwhile gain.

$$N_{C_{max}} = \frac{n(n+1)}{2} - \frac{(n-e)((n-e)+1)}{2} + e \times c \quad (2)$$

Note that the range between $N_{C_{min}}$ and $N_{C_{max}}$ is quite large; the maximum number of gain calculations could also end up being larger by a small factor than the calculations required by AXBAM_NBV. However, every scan at a neighbouring pose increases the probability that when AXBAM_NBV is eventually called, more poses in Q will have zero gain and therefore be discarded. Furthermore, the earlier a pose is discarded from Q, the more calculations can be avoided.

C. AXBAM Algorithm

AXBAM_NBV is used as a module in the NN_NBV algorithm, so a brief overview will be presented here for completeness, and the reader is referred to our original publication for full details [3].

First a set of possible robot configurations, or poses, is generated. This is a uniformly distributed sampling of all possible robot configurations. A discrete number, N_S, of sample angles is taken from each robot joint, and poses are created using every possible combination of these samples (duplicates are discarded). Next, poses are repeatedly chosen and the robot moved to them, until either no poses remain or the information gathered so far is considered sufficient.

The effort required to get from the current pose to the candidate pose is calculated using the function $E()$ which takes in two poses and returns the required joint effort. Effort in this case is defined as being the sum of the absolute difference in each pose's joint angle. Once the gain and effort for each candidate pose have been calculated, any pose with zero information gain is discarded.

To choose the optimal pose $q \in Q_p$, the poses lying on the pareto frontier are found. This allows attention to be restricted to poses that are pareto efficient with regard to

gain and effort. A pose $q_1 \in Q$ is a member of the pareto frontier, if given the current position q_{curr} and map M, there is no pose $q_2 \in Q$ such that,

$$G(q_1, M) \leq G(q_2, M) \ \wedge E(q_1, q_{curr}) \geq E(q_2, q_{curr}) \quad (3)$$

and there is at least one utility function where q_1 is strictly better than q_2.

An optimal pose, q_{nbv}, is chosen from the pareto frontier by selecting the pose with the best ratio of gain over effort. To do this, the function $R()$ is used, which takes in a set of tuples (pose, gain, effort) and returns the best pose in terms of gain over effort. There are other ways the tie between the pareto frontier elements could be broken, such as assigning ranks to values of gain or effort, but the ratio of gain over effort is an intuitive and straightforward way of accomplishing the required selection.

Once q_{nbv} has been chosen, a path to it is calculated from q_{curr} using a RRT planning approach [14]. The robot proceeds through that path, integrating scans into the map at each time step.

The pose selection process is repeated until all poses have either been used or discarded because they offer no gain or are impossible due to obstacles.

Algorithm 3: $AXBAM\_Next\_Best\_View$

Input: $q_{curr} \longleftarrow$Current robot pose , $M \longleftarrow$Octomap, $Q \longleftarrow$Poses under consideration

Output: q_{nbv}, Q

$Q_p, Q_u = V(Q, M)$;
$Candidates = \emptyset$;
$q_{nbv} = \emptyset$;
for $q \in Q_p$ **do**
  $gain = G(q, M)$;
  $effort = E(q_{curr}, q)$;
  if $gain == 0$ **then**
    $Q_p = Q_p - q$;
  else
    $candidate = (q, gain, effort)$;
    $Candidates = Candidates \cup candidate$;
if $Candidates \neq \emptyset$ **then**
  $Candidates = paretoFrontier(Candidates)$;
  $q_{nbv} = R(Candidates)$;
  $Q_p = Q_p - q_{nbv}$;
$Q = Q_p \cup Q_u$;

IV. EXPERIMENTAL RESULTS

Several sets of experiments were performed, including both simulations and using a real robot in the lab.

The simulations were run on an Intel Core 2 Duo (3.00 GHz) with 3.5 GB of RAM in MATLAB using mex files to interface with the C++ Octomap library (v 1.4) [4].

The robot used in the simulation experiments has 5DOF, the number of joint angle samples, N_S, is set to 6, and

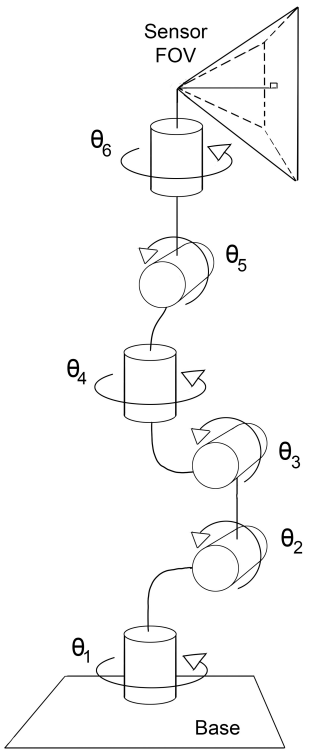


Fig. 1: A 6DOF robot (and sensor field of view)

the sensor is modelled to have a field-of-view and range matching that of the Microsoft Kinect. The sensor is assumed to be mounted on the robot's end effector.

Four simulated environments were used. The first, the Floor environment, used to test the performance of NN_NBV in open spaces, was a flat surface 0.5 by 0.5 meters in size, the robot being placed in the center. The Tunnel environment, to test NN_NBV in simple enclosed environments, was 6 meters long, 1.3 meters wide and 0.8 meters tall. The Window and Plate environments, used to test NN_NBV in more complex environments, were also based around a tunnel, but each contained an obstacle that restricted much of the robot's workspace. For each of the four environments, both algorithms were run until no more information was left to be gathered from any of the poses generated.

Several metrics were recorded in order to compare the two algorithms. In the tables presented here:

- **Time**: number of minutes the algorithms took to run to completion,
- **Information**: number of voxels known to be free or occupied,
- **Gain Calculations**: number of gain calculations performed,
- **End effector movement**: distance travelled in meters by the end effector,
- **Total joint effort**: amount of work done, in radians, by the robot's joints,

Two sets of simulated experiments were run. In the first set, using in turn both the NBV, then the NN_NBV algorithms in all four environments, scans of the environment

TABLE I: Comparisons of the two exploration algorithms in the simulated environments, where scanning was performed along the path as well as at the goal position.

(a) The Floor Environment

Metrics	$AXBAM\_NBV$	NN_NBV	% Diff
Time (min)	14.78	12.49	15.45
Information	386326	395643	2.41
Viewpoints Evaluated	12289	10361	15.69
Effector Movement (m)	24.14	22.53	6.66
Joint Effort (radians)	156.76	121.70	22.36

(b) The Tunnel Environment

Metrics	$AXBAM\_NBV$	NN_NBV	% Diff
Time (min)	12.50	9.29	25.65
Information	47556	46706	-1.79
Viewpoints Evaluated	11267	8265	26.64
Effector Movement (m)	28.91	24.45	15.42
Joint Effort (radians)	169.20	164.97	2.50

(c) The Window Environment

Metrics	$AXBAM\_NBV$	NN_NBV	% Diff
Time (min)	15.75	11.32	28.11
Information	50646	49880	-1.51
Viewpoints Evaluated	14559	9051	37.83
Effector Movement (m)	31.71	27.95	11.84
Joint Effort (radians)	257.29	228.82	11.07

(d) The Plate Environment

Metrics	$AXBAM\_NBV$	NN_NBV	% Diff
Time (mins)	12.50	10.33	17.35
Information	47036	48924	4.01
Viewpoints Evaluated	10834	8353	22.90
Effector Movement (m)	25.68	26.29	-2.35
Joint Effort (radians)	240.59	282.08	-17.25

were only taken at the chosen next best viewpoint. In the second set of tests (Table I), whenever a path to a new viewpoint was generated (as opposed to choosing a nearest neighbour pose), a scan was taken every 5 steps along the path generated by the RRT (or approximately after 5 degrees of work done by the joints).

When only scanning at chosen viewpoints, NN_NBV always gathered more information than AXBAM_NBV. When AXBAM_NBV was made to scan along paths to chosen viewpoints as well as at the viewpoints themselves, NN_NBV gathered at worst 1.79% less information than AXBAM_NBV did and, at best, 4% more. In terms of robot motion and effort, in the simpler environments such as the Floor and Tunnel, and even the Window environment, NN_NBV resulted in a 6 to 15% reduction in distance travelled by the end effector and a 2.5 to 22% reduction in joint effort. In the more complex Plate environment, NN_NBV resulted in only slightly more work done by the robot, with a 2% increase in end effector motion and 17% increase in joint effort.

When not performing scans along end effector paths (on calls to AXBAM_NBV), even the worst case reduction in the number of gain calculations required was over 70%, resulting in 65% less time needed for overall exploration. When scanning was performed along paths (on calls to AXBAM_NBV), the reduction in the number of gain calculations was still over 15% in the worst case, while in the best case, it was reduced by over 37%. The required time for exploration is reduced by over 15%.

TABLE II: Comparisons of the two exploration algorithms using the Denso robot in the lab.

Metrics	$AXBAM\_NBV$	NN_NBV	% Diff
Time (min)	56.60	17.43	69.20
Information	478948	427699	-10.70
Viewpoints Evaluated	8043	1998	75.16
Effector Movement (m)	59.92	24.05	59.86
Joint Effort (radians)	323.02	90.65	71.94

Testing in a real environment was done with a robot manipulator arm with 6DOF (Figure 1) in an open but cluttered environment. The number of joint angle samples, N_S, was set to 4. The joint step size for NN_NBV was 10 degrees. A Kinect sensor was attached to the arm's end effector. Both the NN_NBV and AXBAM algorithms were used to explore the environment surrounding the robot and collect the largest amount of information possible given the range of the sensor and reach of the arm. When moving to points chosen by the AXBAM_NBV function, a scan was taken every 10 steps along the path.

Shown in Table II are the results comparing both algorithms when exploring the lab. NN_NBV was shown to be faster while collecting approximately the same amount of information, and collecting more information over time. NN_NBV also resulted in more efficient robot motion. Shown in Figure 3 is the resulting voxelised representation of environment collected by the robot using NN_NBV.

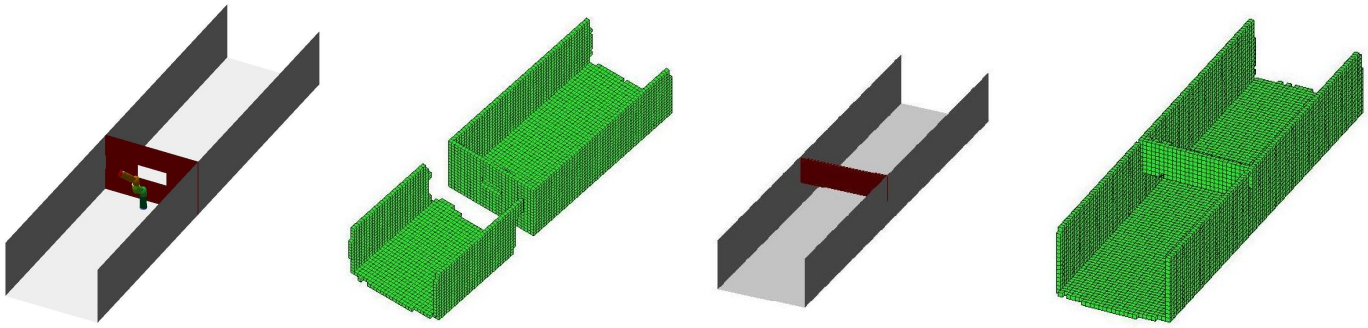
From both the simulated and real experiment results, NN_NBV outperforms AXBAM_NBV in terms of information gathered over time.

V. CONCLUSIONS AND REMARKS

An extension to an existing frontier-based exploration algorithm has been presented that can be adapted to other robots and scenarios for exploring 2D and 3D environments, particularly for robots involving manipulators equipped with one or more sensors.

It was demonstrated that considering only the closest viewpoints for as long as possible can help significantly reduce the number of viewpoints that have to be evaluated while still offering comparable and even superior results in information gathered and an economy in robot motion. This results from the fact that planning further ahead yields diminishing improvements, which become eclipsed by planning costs. Our results also confirm that in each test, there was a significant reduction in the number of gain calculations and, as a result, in the overall computation time.

The Nearest Neighbour exploration strategy also exhibits a reduction in overall distance travelled by the end effector and the total amount of work done by the robot's joints. Combined with the reduced computation time, this results in a reduction of the total time required for the robot to explore the environment.



(a) (b) (c) (d)

Fig. 2: Simulated Environments and resulting maps, ceilings of each representation have been removed for clarity and only the occupied voxels of the maps are shown: (a) Window Environment (with robot inserted for scale), (c) Plate Environment with an obstacle that the robot can reach under, (b) and (d) the resulting maps.

Options to further improve the efficiency of NN_NBV will be investigated by testing the different manners in which the nearest neighbour poses could be generated to ensure a more even spread of possible information gain between neighbours.

REFERENCES

[1] C. Potthast and G. S. Sukhatme, "Next best view estimation with eye in hand camera," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS): The PR2 Workshop*, 2011.

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

[3] G. Paul, S. Webb, D. K. Liu, and G. Dissanayake, "Autonomous robot manipulator-based exploration and mapping system for bridge maintenance," *Robotics and Autonomous Systems*, vol. 59, no. 78, pp. 543–554, 2011.

[4] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," *In Proc. of the ICRA workshop*, 2010.

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

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

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

[8] R. Shade and P. Newman, "Choosing where to go: Complete 3d exploration with stereo," *Robotics and Automation, IEEE International Conference on*, pp. 2806–2811, 2011.

[9] M. Morales, R. Pearce, and N. Amato, "Analysis of the evolution of c-space models built through incremental exploration," *Robotics and Automation, IEEE International Conference on*, pp. 1029–1034, 2007.

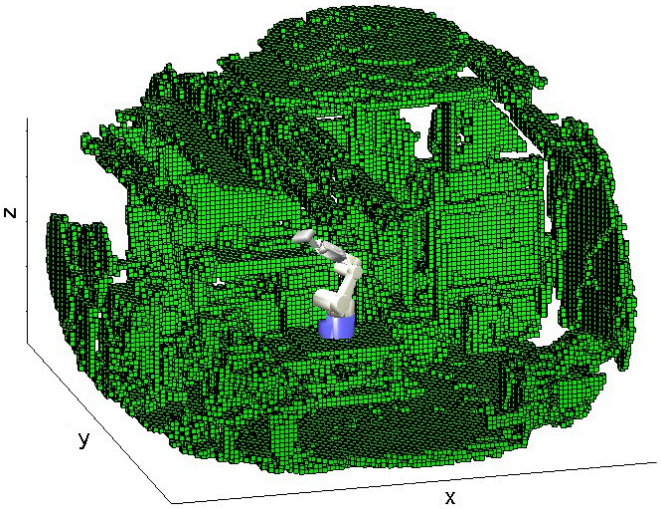
[10] B. Adorno and G. Borges, "iARW: An incremental path planner algorithm based on adaptive random walks," *Intelligent Robots and Systems, IROS. IEEE/RSJ International Conference on*, pp. 988–993, 2009.

[11] Y. Huang and K. Gupta, "An adaptive configuration-space and workspace based criterion for view planning," *Intelligent Robots and Systems, IROS. IEEE/RSJ International Conference on*, pp. 3366–3371, 2005.

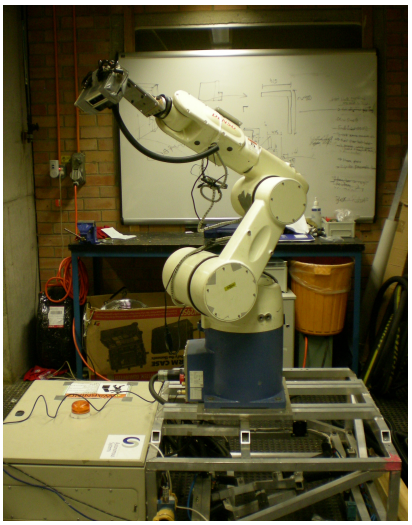
[12] K. Rayner, "Eye movements in reading and information processing: 20 years of research." *Psychological Bulletin*, vol. 124(3), pp. 372–422, 1998.

[13] J. M. Henderson, "Human gaze control during real-world scene perception," *Trends in Cognitive Sciences*, vol. 7, no. 11, pp. 498 – 504, 2003.

[14] M. Clifton, G. Paul, N. Kwok, D. Liu, and D. L. Wang, "Evaluating performance of multiple RRTs," *Mechtronic and Embedded Systems and Applications. MESA. IEEE/ASME International Conference on*, pp. 564 –569, 2008.



(a)



(b)

Fig. 3: Experimental results with Denso robotic arm: (a) the resulting voxel map of the lab environment (occupied voxels only), and (b) the experimental setup in the lab.