Exploring in 3D with a Climbing Robot: Selecting the Next Best Base Position on Arbitrarily-Oriented Surfaces

Phillip Quin, Gavin Paul, Alen Alempijevic, Dikai Liu

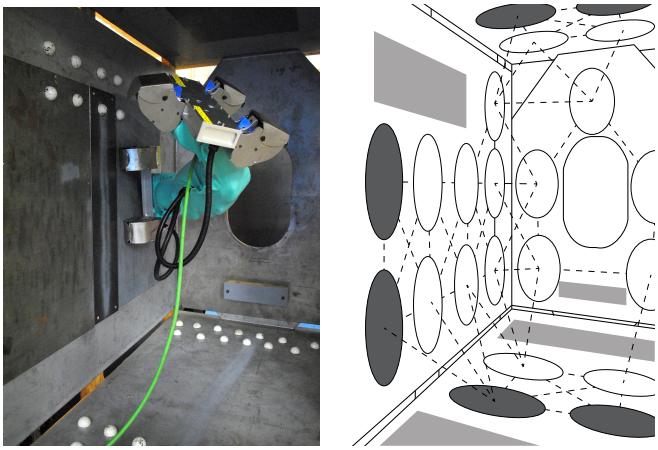
***Abstract*— This paper presents an approach for selecting the next best base position for a climbing robot so as to observe the highest information gain about the environment. The robot is capable of adhering to and moving along and transitioning to surfaces with arbitrary orientations. This approach samples known surfaces, and takes into account the robot kinematics, to generate a graph of valid attachment points from which the robot can either move to other positions or make observations of the environment. The information value of nodes in this graph are estimated and a variant of A* is used to traverse the graph and discover the most worthwhile node that is reachable by the robot. This approach is demonstrated in simulation and shown to allow a 7 degree-of-freedom inchworm-inspired climbing robot to move to positions in the environment from which new information can be gathered about the environment.**

I. Introduction

Most robot exploration strategies make use of the fact that the robot is moving through an environment that can be represented in 2D, even when the robot operates in a 3D world [1], [3], [7], [13], [15], [16]. This reduction in dimensions is key in reducing computation. For example, a robotic vacuum cleaner might represent an office floor as a 2D plane, since only objects on the floor are within its scope of concern. When exploring the 2D plane, all positions on the ground plane are potential locations from which the robot can make an observation. Freespace areas are often implicitly assumed to be space that the robot can move through. As a result, path planning, particularly for a holonomic robot, becomes straightforward. These assumptions do not hold for a robot tasked with climbing through a 3D structure.

Rather than Cartesian space, some exploration strategies operate in the robot's configuration space (C-space). These strategies are well suited to robot manipulators that operate from a fixed base position [2], [4], [9], [19], or the base of the manipulator moves on one plane (e.g. the floor).

A climbing robot which must climb the internal structure of bridges for inspection [10], such as the one shown in Fig. 1a must be able to make use of surfaces in any orientation; not only in determining where it can possibly move to, but also in determining what it might see from a particular location. Since the robot can move through 3D space but is bound to surfaces, path planning is more complex. While an exploration strategy for such a robot system could be constructed almost entirely using Rapidly exploring Random Trees (RRT) [18] similar to work by [5], this would involve a large number of dimensions. At least 4 and potentially as many as 9 dimensions are required to describe the robot's position and orientation, and further dimensions are required to represent the robot's degrees of freedom in configuration space. Given that in 3D physical space, only the slices of the space that represent the surfaces will be valid, this means the vast majority of possible points in the n-dimensional space will be invalid. An approach based on random sampling is thus unlikely to be tractable. Similarly, evaluating each position on a surface as a candidate for the next observation of the environment, as done in the 2D case by [1], is unlikely to be tractable in the 3D case.



(a) (b)

Fig. 1: (a) A climbing robot navigating a 3D environment. (b) Example attachment points (discs) on a map of known surfaces with edges between them (dashed lines). Light grey surfaces are unsafe to attach to, dark discs are sufficiently close to the frontier set that they belong to the set of candidates

This paper presents a novel graph-based approach to autonomous exploration which does not limit the robots motion to a single 2D plane, and does not assume that the robot is holonomic for path planning purposes. Instead of uniformly sampling in the robot's configuration space, computational complexity is reduced by generating candidate robot configurations from surfaces in the map which can then be connected in a directed graph of edges and evaluated for potential information gain. Interleaving the increasingly expensive but more accurate information estimation and cost functions further reduces computational complexity by progressively filtering candidate configurations.

The approach begins by sampling all surfaces in the environment to determine candidate robot attachment points represented as nodes in the directed graph. The respective

This work is supported by Roads and Maritime Services, NSW.

The authors are with the Centre for Autonomous Systems at the University of Technology Sydney, Australia.
Phillip.Quin@uts.edu.au.

strengths of A* and RRT planners are leveraged to determine edges of the graph. A modification to traditional A* is proposed that aims to reduce computational complexity by simultaneously searching for multiple goals. Edge validation is done twofold, initially using the existence of a robot kinematic configuration between nodes. Subsequently, a RRT planner is used to validate existence of a complete trajectory between nodes. As a result, the proposed approach demonstrates continuous exploration of the world, progressively sensing regions that are non-observable from the potential viewpoints of a robot in any configuration at the current location. The exploration approach is described in Section II and results from simulations are presented in Section III. Conclusions and future work are discussed in Section IV.

II. Methodology

This paper assumes that some region of the environment is known to the robot as a result of past observations. The robot's knowledge of the environment is stored in a map of free and occupied space, $\mathcal{M}_o$, and a map of known surfaces, $\mathcal{M}_s$. This approach further assumes that the set of frontiers, $\mathcal{F}$, the boundaries between known and unknown space in $\mathcal{M}_o$, has been created.

The robot is assumed to be affixed to a surface by its "base", which can be described by a homogeneous transform, ${}^0T_b$. The robot is able to make a series of observations whilst its base is affixed to a surface.

Modelling the dynamics of this robot is out of the scope of this work, and some details can be found in [17]. The effects and impacts of robot dynamics are assumed to be incorporated as part of determining safe attachment points and path-planning which are referred to later in the paper.

Determining where the robot should move to next to make a new series of observations of the environment involves several steps:

A) Determine attachment points
B) Estimate candidate attachment point information
C) Selectand move to the next best attachment point

A. Determining Attachment Points

M_s is uniformly sampled to generate a useful distribution of valid attachment points, A, for the climbing robot. An example of a simple sampling strategy is shown in Fig. 1b. Let the sampling policy be denoted, $s()$, and let the set of valid attachment locations, A be returned by $s(M_s)$.

The way in which sampled points are used to create valid attachment points, and how these attachment points are represented, depends on the mechanism by which the robot adheres to the surface. Techniques exist to find footpad placement positions in rough terrains [6]. If the robot has a magnetic footpad that can be abstracted as a disc such as [8], [14], then this disc abstraction can be used with the sampled point as the center. Fig. 2a shows how the disc, along with an orientation and normal can represent a placement of the robot's footpad. The normal is the cross product between two unit vectors created from the sample point in the direction of 2 different nearby sample points. A predetermined set of orientations vectors (dictating footpad direction when attached to a surface) are combined with the point-normal pair to create a set of attachment points which are checked for validity. If there are m point-normal pairs sampled from the M_s, and n predetermined orientations, then up to $m \times n$ attachments points will be added to A.

The orientation of a specific attachment point, a in A affects which nearby attachment points are reachable by the robot, what observations the robot can make from a, and the adhesion safety of the robot at a (based on the attachment mechanism and the kinematics of the robot). a can be described as a tuple containing a point $\mathbf{p}$, normal $\mathbf{n}$, orientation $\mathbf{o}$, and safety value $\mathbf{v}$. Unsafe attachments, with safety values below a threshold τ_s, are culled from A.

Nodes are connected by creating edges. Given attachment nodes a_i and a_j, an edge, $e_{i,j}$, is created from a_i to a_j if the distance, d between them is within the range $[d_{min},d_{max}]$, where d_{min} and d_{max} are the smallest and largest transition the robot can make, respectively. If the trajectory from a_2 to a_1 is the reverse of the trajectory from a_1 to a_2, then a single undirected edge is sufficient to link two attachment nodes, otherwise separate directional edges are needed.

If attachment nodes belong to distinct robot states, such as front-footpad-attached and rear-footpad-attached, these states preclude certain attachment nodes being connected. This information can be used to quickly determine whether an edge between two attachment nodes can be ignored. For example, consider an inchworm-inspired robot which has two footpads and moves with a flipping gait [14]. In this case, attachment nodes belong to two distinct states with either (or both) of the front and/or rear footpad attached, and cannot have an edge with an attachment point of the same state.

Associated with each edge, $e_{j,k}$ is a robot configuration $\mathbf{q}_{j,k}$ in which the robot is attached to both a_j and a_k (i.e. the robot is "straddling" a_j and a_k) as shown in Fig. 2b. Initially, $\mathbf{q}_{j,k}$ is unknown (i.e. is ungenerated). Each edge, $e_{j,k}$ is also associated with a path $\mathbf{p}_{i,j,k}$ describing a sequence of configurations required to move from $\mathbf{q}_{i,j}$ to $\mathbf{q}_{j,k}$, assuming the robot came from $e_{i,j}$. These paths are initially empty.

After an initial set, A has been generated on M_s, subsequent map updates only require resampling the subset of M_s that has changed. After each sequence of observations from an attachment point, the updated region, m_s, can be approximated as the portion of M_s within a sphere centered on the robot's current attachment node and with radius equal to the length of the robot, r_l, plus the trustworthy range of the sensor, r_s.

Each a in A that is within the sphere needs to be evaluated to check it is still valid after the map update. The sampling function, $s(m_s)$ will generate the set of attachment nodes, A_s. Attachment nodes in A_s that are similar to any attachment in A are discarded. Remaining attachment nodes in A_s are added to A, and edges are created between nodes in A_s and any other attachments in A as required.

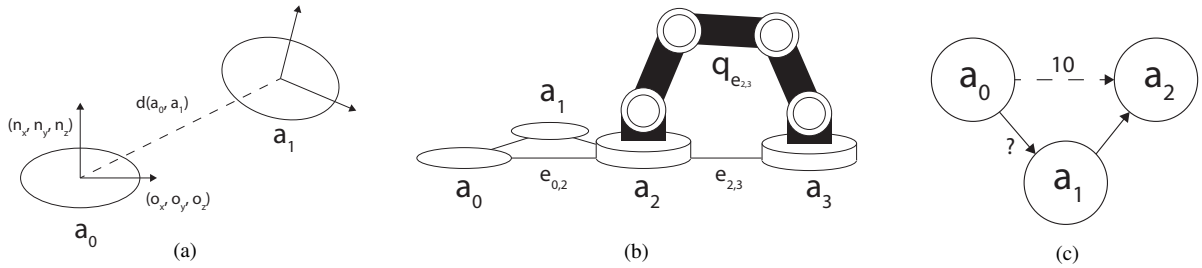


Fig. 2: Updating the graph of attachment positions. (a) Two attachment points. (b) Attachment nodes and edges between them. The robot is shown in configuration $q_{2,3}$, which "straddles" attachment nodes a_2 and a_3. (c) Attachment points and the costs of moving from a_0, to a_1 and a_2.

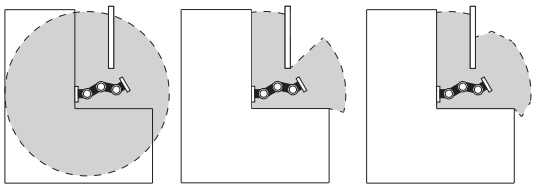


Fig. 3: Several methods for estimating information: Left: unknown in sphere, Middle: raycasting from robot base [1], Right: raycasting from sensor transforms resulting from sampled robot configurations [11].

B. Estimating Candidate Attachment Point Information Gain

We define a function, $H(a)$ which estimates the information score of an attachment point, a. Fig. 3 shows three example $H(a)$ information estimate functions. The simplest estimate of information around a candidate attachment point is to count the unknown voxels within a sphere of radius $r_l + r_l$. The most expensive, but more accurate method is to count the unknown voxels that would be covered by a representative sampling of observations resulting from different robot configurations at that attachment position. Calculating $H(a)$ for each $a \in A$ would be computationally expensive. Therefore only a randomly selected portion λ of attachment points within r_l+r_l of any frontiers are evaluated.

C. Selecting and Moving to the Next Best Attachment Point

The set of candidate attachment points for which an information value has been estimated is denoted $\mathcal{C}$. To determine which node in $\mathcal{C}$ is the Next Best Attachment position (NBA), nodes in $\mathcal{C}$ need to be ranked in terms of utility; a measure of a candidate's desirability as the next location from which exploration should occur.

Assuming the robot is currently at attachment point, a_0, the utility value, $U(a,a_0)$, of a, is a function of its information score, $H(a)$, and its cost, given by a function $C(a,a_0)$. The cost of a is the sum of the costs associated with the edges that form a connected sequence to it from a starting node. The cost of moving from one node in the graph A to another is bounded by a maximum value.

The cost of a sequence of edges is therefore given by three components: 1) u, the number of edges without path information, 2) k, the number of edges with path information, and 3) e, the sum of the cost of known path information. The cost of reaching the candidate attachment a_j given a starting point of a_i is $C(a_j|a_i) =< u, k, e >$. The cost of an edge is either simply $(1, 0, 0)$ (for an unknown step), if there is no path information for the desired route, or $(0, 1, C(p))$ (for a known step) and the cost of the known path, p. If the robot's current position is a configuration that straddles the nodes a_3 and a_0 then the path p to move to a position straddling a_0 and a_1 is $p_{3,0,1}$.

Summing the cost of two sequences is done by summing each individual component. If $C(a_1,a_0) = (u_1, k_1, e_1)$, and $C(a_2,a_1) = (u_2, k_2, e_2)$, then:

$$C(a_2,a_0) = (u_1 + u_2, k_1 + k_2, e_1 + e_2) \tag{1}$$

Fig. 2c shows as example where the cost from a_0 to a_1 is $(1, 0, 0)$. The dashed line shows the trajectory cost of an end effector moving from a_0 to a_2, while the other end effector is at a_1; this trajectory is associated with the edge from a_1 to a_2. The cost from a_1 to a_2 is therefore $(0, 1, 10)$. Thus, the cost to get to a_2 given a starting position of a_0 is $(1, 1, 10)$.

A cost (u_1, k_1, e_1) is considered better than (u_2, k_2, e_2) if:

$$(u_1 + k_1 < u_2 + k_2) \wedge (u_1 < u_2) \wedge (e_1 < e_2) \tag{2}$$

This ensures that:

1) Shorter sequences are preferred over longer ones.
2) Given sequences of equal length, those with higher proportion of known paths are chosen.
3) Given equal number of known paths, the smallest cost paths should be chosen.

This ordering reflects the optional path strategy of a robot, such as the climbing robot in our scenario, where the actions involved in attaching and detaching to the surface take significant periods of time.

Determining the cost to reach each candidate would be computationally expensive and unnecessary. Instead, a variant of A* is used to find the shortest path from the robot's current position, s to the most worthwhile candidate. The variant of A* differs from regular A* in the following ways:

1) All candidates with evaluated information scores above a given threshold are simultaneously given as goals $g_0, g_1, \ldots, g_n$.
2) The heuristic score for node a_e is:

$$\max_{i \in [1,\ldots,n]} U(g_i,s) \tag{3}$$

3) Given the robot's starting position, s, the node currently being expanded by A*, a, the node whose

heuristic score is being calculated, a_h, the utility value for a goal node, g is:

$$U(g,s) = \frac{H(g)}{(C(a,s) + C(a_h,a) + hC(g,a_h))^3} \quad (4)$$

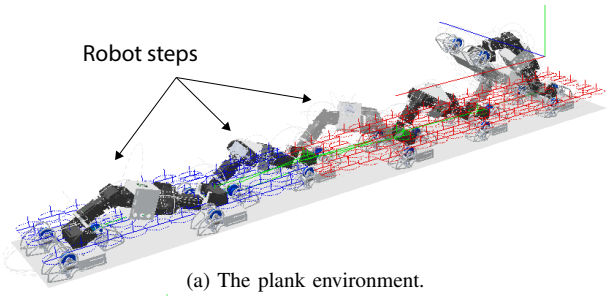
where $hC(a,s)$ is a heuristic cost estimated as the Euclidean distance to the target divided by the maximum step length of the robot. The tuple values represented by the $C()$ function are converted into scalar values. In this paper, this scalar value of $C()$ was the sum of the u and k values of the tuple. Whilst other functions could have been chosen, the run-time of A* in this paper is dominated by the cost of discovering the configuration, $q_{i,j}$ associated with $e_{i,j}$. The number of edges that might have configurations evaluated by A* is proportional to the volume of a sphere whose radius is the number of edges from the robot's current position to the selected candidate. The function describing the volume of a sphere is a third degree polynomial; the radius of the sphere is cubed. The utility function is therefore selected to include the inverse to penalise expanding the sphere of edges evaluated by A*.

4) When a node i is expanded, the node j at the other end of each edge, $e_{i,j}$ is only added to the open set (of nodes to consider next) if there is a "straddling" configuration, $q_{i,j}$, associated with the edge. If there is no such configuration, then a configuration is generated. If no valid configuration can be found, then the edge is deleted, and the node at the other end is not added to the open set.
5) If node a is expanded and it is one of the goals, but its heuristic score is not equal to its utility, then A* continues as if this was not a goal node, and this node's heuristic score is set to be its own utility, and re-added to the open set.
6) A* terminates when it is expanding a candidate node whose best score is itself, and not a different goal.

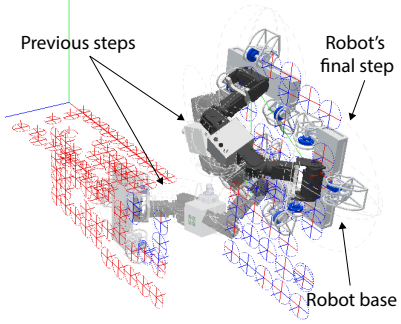
Once A* finds a sequence of nodes and edges from the robot's current position to a target node, then a path planner such as RRT which takes the robot's kinematics into account [17] is used to move the robot to the new location.

If path planning is successful, the robot moves to the new location and makes new observations of the environment. If A* or path planning is unsuccessful, then the edge for which no trajectory was found is either quarantined (marked as unusable) or deleted. An edge is quarantined if the path planner would have found a solution if unknown space wasn't considered as an obstacle. If discounting unknown space still doesn't result in a trajectory, then the edge is deleted. Edges are un-quarantined after new observations have been made of the environment. When A* is unable to find a path to a goal, λ percent of the remaining attachment points that have no information scores are evaluated and added to the goal nodes given to A*.

The exploration approach terminates when all candidates have had their information scores evaluated and A* still fails to reach a target.



(a) The plank environment.



(b) The beams environment.

Fig. 4: Results of Simulation 1. The robot is shown in the start and end positions. Attachment points are shown as red and blue discs. Blue discs are candidate next best attachment positions. Lines from the center of the discs represent the normals and orientations. Multiple discs occupy the same position coordinates.

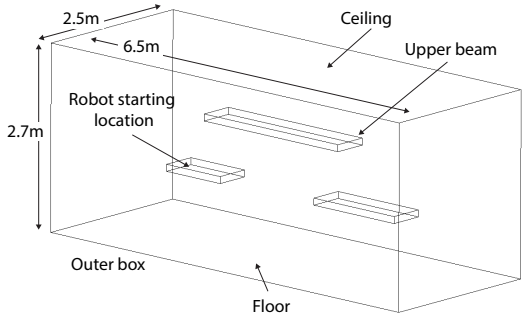


Fig. 5: Wire frame of the environment in Simulation 2.

III. Results

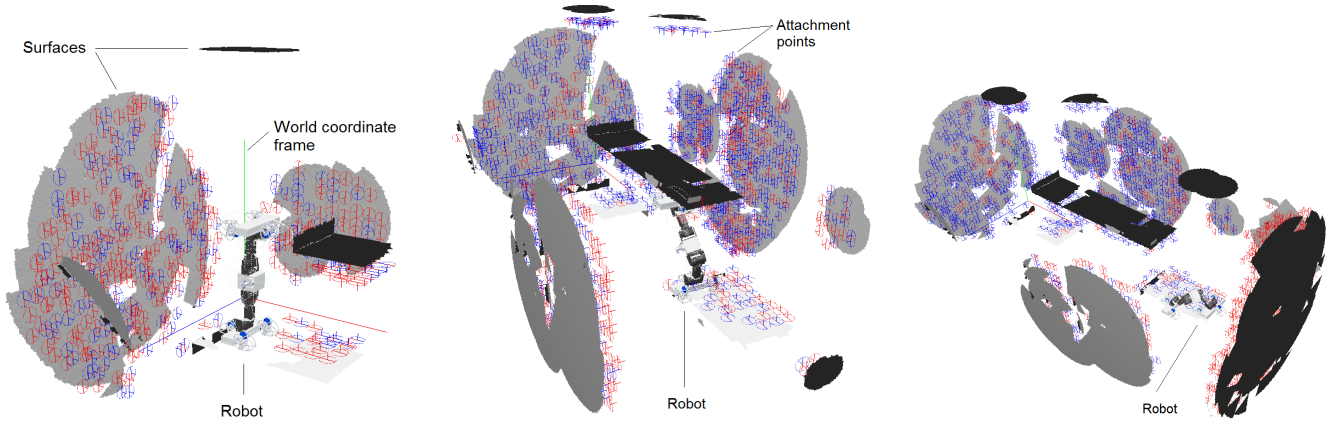
The approach presented in this paper is demonstrated using several simulated environments in which a 7DOF robot [14] must select the next best attachment position. The code was implemented in C++ and run on a desktop computer with an Intel i7-4770 3.40GHz processor with 16GB RAM.

Two sets of simulations were performed.

A. *Simulation 1*

The selection of the Next Best Attachment position was tested in several environments. In each environment, some surfaces and regions where already known, as was the set of frontiers. The approach used to predict information gain $H(g)$, given a node g, is the second approach in Fig. 3: counting unknown voxels traversed when raycasting in a sphere of radius $r_l + r_s$ from the robot's base.

In the environments shown in Fig. 4, the surfaces were sampled to form a grid with a resolution of 0.1 meters. Edges were created between nodes in the graph if they were between 0.4 and 0.6 meters apart. In the simple plank



(a) Nodes shown as discs near the surface. Those in blue have had $H()$ evaluated.

(b) The robot transitioning between the upper beam and the lower right beam.

(c) The robot in its final resting place after exploration has ended.

Fig. 6: Visualisations of Simulation 2.

environment, two possible orientation were set. In the beams environment, four possible orientations were set to allow motion in each direction. Attachment points are shown as discs. Blue discs are candidate Next Best Attachment positions. The normals and orientations of attachment points are shown as lines from the center of the disc. In both environments, a path is successfully found between the start position and a goal position. The robot took 2.3 minutes to determine and plan a path to the NBA in the plank environment, and a further 3.8 minutes to move to the NBA, with robot motion being simulated at real speeds. The same tasks took 8.2 and 3.7 minutes respectively in the beam environment.

B. *Simulation 2*

Simulation 2 involved repeated selection of the NBA, interleaved with observations of the environment (Local Exploration). The exploration strategy used for Local Exploration was the Nearest Neighbour Exploration Approach in [12], suited to manipulators with a fixed base position. Fig. 5 shows the environment used, consisting of three beams suspended within a larger box such that the robot could physically transition between the beams but would be unable to reach the outer surface. The sensor was modelled on a PrimeSense depth camera, with a field of view of 43×57 degrees, but with an intentionally limited range of 1 meter, to force the robot to move to new base locations in order to explore the environment.

This simulation was performed several times, but a single instance is presented here in detail as it is representative. Care needs to be taken in selecting the density of the surface sampling and the number and type of resulting attachment nodes. Too few nodes might result in a poor NBA, or in the inability of the robot to find paths between nodes. Large sets of nodes will result in higher computation times.

The robot is initially placed in the environment and the only known information is that a small volume of space around the robot is known to be freespace. Local Exploration takes place, then the NBA is determined using the approach presented in Section II. As part of this process, 1 457 new attachment nodes are created (representing both "feet" and "hand" positions), and 28 088 edges. There are therefore only 728 candidate nodes, since exploration cannot take place from a "hand" node.

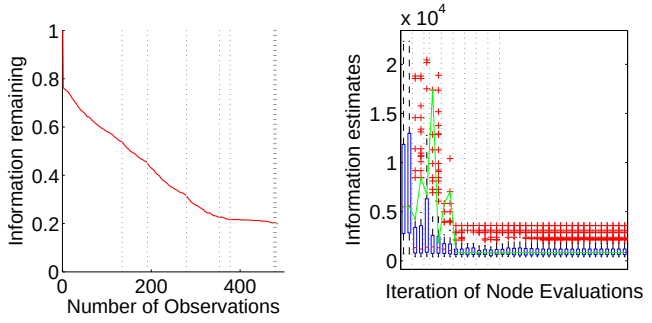
151 of the 728 candidates ($\lambda = 20\%$) have their information scores evaluated. The evaluated nodes are shown as blue discs in Fig. 6a, while unevaluated nodes are marked as red discs. The line perpendicular to the disc is the normal, and the line from the center of the disc along the surface of the disc is the orientation of the node. The nodes' positions are relative to the robot's footpad coordinate frame, and are therefore a distance off the surface.

A* is then performed using these evaluated candidates as goal nodes. A sequence of edges is found, comprising of at least two edges. The path planner is given this sequence of nodes and edges and attempts to find a valid trajectory to those target nodes from the robot's current position. Having successfully arrived at a new position in the environment from which observations can be made, the Local Exploration stage begins again. Fig. 6b shows the robot moving from the second to the third beam.

After the 10th instance of Local Exploration, the robot is in the position shown in Fig. 6c. Nodes are evaluated, A* is performed, and no path is found to any candidate attachment point. Although A* does not find a path, 20% of the remaining unevaluated nodes are evaluated and A* is run again. Each subsequent run of A* is faster, since a higher proportion of edges will have been evaluated by previous iterations of A* and either been assigned a robot configuration that "straddles" the nodes, or the edge will have been deleted. This process of evaluating nodes and then performing A* repeats until no node remains unevaluated, and A* still finds no path from the robot position to any desirable node, at which point exploration terminates.

There are a total of 12 586 nodes in the graph, 793 228 edges, and 485 observations were made as part of 10 occurrences of Local Exploration with the robot affixed to different locations in the environment.

The total number of voxels within the larger box is its volume ($6.5m \times 2.5m \times 2.7m$), divided by the volume of a single voxel ($0.05^3 m^3$), or approximately 351 000 voxels.



(a) (b)

Fig. 7: (a) Estimated information remaining over time. (b) Box plot of estimated information of evaluated nodes at each iteration (blue: 25-75th percentile, red: outliers). Green shows the selected attachment point. Vertical lines separate occurrences of Local Exploration.

This is an overestimate of the total information that should be able to be captured by the robot. Fig. 7a is a plot of the amount of unknown information within the outer box over time, showing the effectiveness of the exploration approach in observing the majority of the environment. At the end of exploration, only 70 064 unknown cells remain within the outer box; 80% of the volume has been explored.

Fig. 7b shows the estimated information values of each evaluated node at each timestep that a Next Best Attachment point was selected. The fact that the estimated information values of each evaluated node follows the same trend as that of the information in Fig. 7a validates its use as an information metric.

Exploring 100% of the volume is impossible. The sensor range is such that the corners of the outer box are unobservable from any position on the beams. The configuration of the beams also means that the robot cannot, from one side, completely reach around to freely observe all of the regions of space on the other side of the beam.

The total time taken was 161 minutes: 61 minutes spent for 10 instances of Local Exploration, 0.4 minutes updating the graph, 2.6 minutes updating the set of frontiers, and 22 minutes were taken to update and estimate the information value of the candidate attachment points. Graph operations, performing m-A*, and path-planning using RRT, required a total of 35 minutes. The robot took the remaining 38 minutes to complete the planned trajectory (the robot simulation included moving the robot at realistic speeds).

IV. Conclusion

This paper has presented a graph-based exploration approach that allows a climbing robot—able to adhere to surfaces with arbitrary normals—to choose a new attachment position which enables making high information gain observations in such an environment. The approach, which effectively combines the relative strengths of A*—modified to search simultaneously for multiple goals—and RRT, has been demonstrated in simulation and is shown to enable the robot to select a new base location that allows new information to be collected whilst also minimising movement. The approach has been further shown to enable a robot to explore the majority of several simulated environments. While the approach has been demonstrated on a 7 DOF robot, the framework is suitable to any climbing robot that must remain attached to the surface throughout its motion.

Work is under way to validate this approach in real environments with the physical robot.

References

[1] P. Blaer and P. Allen, "Data acquisition and view planning for 3-d modeling tasks," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 2007, pp. 417–422.

[2] D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi, "Autonomous navigation and exploration in a rescue environment," in *Safety, Security and Rescue Robotics, Workshop, IEEE International*, 2005, pp. 54–59.

[3] 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*, 2011, pp. 351–356.

[4] L. Freda and G. Oriolo, "Frontier-based probabilistic strategies for sensor-based exploration," in *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on*, 2005, pp. 3881–3887.

[5] L. Freda, G. Oriolo, and F. Vecchioli, "Sensor-based exploration for general robotic systems," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 2008, pp. 2157–2164.

[6] D. Kanoulas and M. Vona, "Bio-inspired rough terrain contact patch perception," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2014, pp. 1719–1724.

[7] M. Lozano Albalate, M. Devy, J. Miguel, and S. Marti, "Perception planning for an exploration task of a 3d environment," in *Pattern Recognition, Proceedings. 16th International Conference on*, vol. 3, 2002, pp. 704–707.

[8] A. Mazumdar and H. H. Asada, "Mag-foot: A steel bridge inspection robot," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 2009, pp. 1691–1696.

[9] G. Paul, P. Quin, A. To, and D. Liu, "A sliding window approach to exploration for 3d map building using a biologically inspired bridge inspection robot," in *CYBER Technology in Automation, Control, and Intelligent Systems, IEEE International Conference on*, 2015, pp. 1097–1102.

[10] G. Paul, P. Quin, A. W. K. To, and D. Liu, "A sliding window approach to exploration for 3d map building using a biologically inspired bridge inspection robot," in *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference on*, June 2015, pp. 1097–1102.

[11] G. Paul, "Autonomous exploration and mapping of complex 3d environments by means of a 6dof manipulator," Ph.D. dissertation, University of Technology Sydney, 2010.

[12] P. Quin, G. Paul, D. Liu, and A. Alempijevic, "Nearest neighbour exploration with backtracking for robotic exploration of complex 3d environments," in *Australasian Conference on Robotics and Automation (ACRA), Proceedings of*, 2013.

[13] R. Shade and P. Newman, "Choosing where to go: Complete 3D exploration with stereo," in *Robotics and Automation (ICRA), Proc. IEEE International Conference on*, 2011, pp. 2806–2811.

[14] P. Ward, D. Liu, K. Waldron, and M. Hassan, "Optimal design of a magnetic adhesion system for climbing robots," in *Climbing and Walking Robots (CLAWAR), International Conference on*, 2013.

[15] S. Wirth and J. Pellenz, "Exploration transform: A stable exploring algorithm for robots in rescue environments," in *Safety, Security and Rescue Robotics (SSRR), IEEE International Workshop on*, 2007, pp. 1–5.

[16] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Robotics and Automation (CIRA), Proc. Symp. IEEE Int Computational Intelligence in*, 1997, pp. 146 – 151.

[17] C. Yang, G. Paul, P. Ward, and D. Liu, "A path planning approach via task-objective pose selection with application to an inchworm-inspired climbing robot," in *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2016.

[18] A. Yershova, L. Jaillet, T. Simeon, and S. LaValle, "Dynamic-domain rrts: Efficient exploration by controlling the sampling domain," in *Robotics and Automation (ICRA), Proc. of the IEEE International Conference on*, April 2005, pp. 3856–3861.

[19] Y. Yu and K. K. Gupta, "C-space entropy: A measure for view planning and exploration for general robot-sensor systems in unknown environments." *I. J. Robotic Res.*, vol. 23, no. 12, pp. 1197–1223, 2004.