Key Feature-based Approach for Efficient Exploration of Structured Environments

Gavin Paul, Phillip Quin, Chia-Han Yang, Dikai Liu

Abstract— **This paper presents an exploration approach for robots to determine sensing actions that facilitate the building of surface maps of structured partially-known environments. This approach uses prior knowledge about key environmental features to rapidly generate an estimate of the rest of the environment. Specifically, in order to quickly detect key features, partial surface patches are used in combination with pose optimisation to select a pose from a set of nearest neighbourhood candidates, from which to make an observation of the surroundings. This paper enables the robot to greedily search through a sequence of nearest neighbour poses in configuration space, then converge upon poses from which key features can best be observed. The approach is experimentally evaluated and found to result in significantly fewer exploration steps compared to alternative approaches.**

I. Introduction

Advances in robotics and sensor technology increasingly enable complicated tasks to be automated, such as inspecting the health of structures [1], [2]. In order to safely and reliably perform robotic tasks in complex three dimensional (3D) environments, a geometrically accurate and reliable map is required - particularly when a robot must plan the precise motions for a given workspace on-line. A robot that climbs around on surfaces in the environment [1], must be able to adhere to the 3D surfaces, whilst determining future surface locations where it can attach, and plan to traverse these surfaces. The need to adhere to surfaces makes planning the stepping and traversal motions complex. In the case of partial and spurious surface data, foot placements must be accurately determined so as to avoid certain areas and collisions. Planning can be significantly aided by augmenting sensor data with prior knowledge about the structure of an environment, such as by using surface templates.

Several algorithms exist for building a map of an environment with a robot. Maximal C-space Entropy Reduction (MER) involves finding the Next Best Viewpoint (NBV) that most reduces C-space entropy [3]. Other approaches greedily select a NBV based on estimates of how much information can be seen from a set of candidate viewpoints [4]–[6].

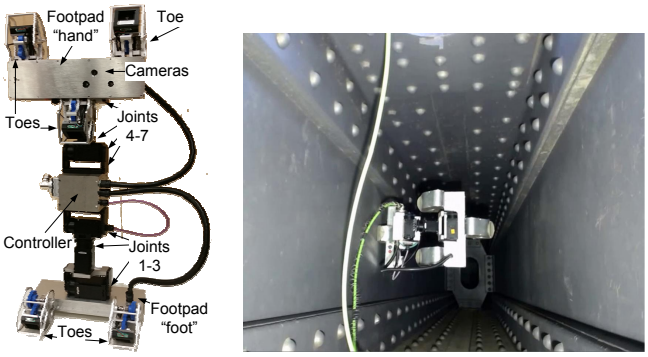
When a 3D map must be generated from multiple observations, registration is generally required so that the overlapping regions of different observations can be fused together. Noisy and incomplete sensor data may mean that even after exploring an environment, the geometry cannot be determined with sufficiently high accuracy. In these cases, data is registered and overlapping parts of several views are fused together. However, a robot with the flexibility to traverse complicated surfaces, and carry its own adhesion mechanism, has been found to be subject to deflection in its links caused by gravity [1], [7], making sensor data taken from multiple poses more complicated to fuse than for a rigid industrial robot manipulator. Noisy and incomplete sensor data, combined with inaccuracies in the robot model, or flexible joints/links, mean that even after exploring an environment the geometric map may be unusable.

Algorithms exist for scanning parts to generate CAD models [8]. These focus on "outside-looking-in" viewpoint planning [9], where the model to be acquired is bounded by a closed volume and scans are taken from a limited set of poses. The unseen areas of the part are analysed and the sensor is moved, by means of the eye-in-hand industrial robot such that the new areas are observed. Intuitively, there is a difference when attempting to inspect and map a surrounding environment where the robot and sensor are "inside-looking-out". An alternative strategy is to identify the location of key features in the environment, to which a template can be fitted, encapsulating prior knowledge of the environment [6], [10].

The scenario addressed in this paper involves a climbing robot with 7 Degrees Of Freedom (DOF) in a caterpillar-inspired configuration [1], which must explore and navigate through the interior of a bridge's steel box girder (forming a tunnel environment). When the climbing robot is positioned within the tunnel and such that several surfaces are visible, it must have the ability to quickly determine the position and orientation of key features in the environment. In this environment a key feature is a view that contains multiple surfaces and surface intersections. The ideal case is to observe all four planes of the tunnel simultaneously [6]. This can be achieved by analysing the partial features in the environment and generating robot motions that allow subsequent observations to improve the view of key features.

The main contribution of this paper is an approach to exploration which finds safe, viable poses that allow a robot to maximise the number of observable surfaces. This approach is used by climbing robot system required to identify a tunnel which must be traversed. This paper is organized as follows, Section II describes the process of partial map generation, surface feature identification and template matching. It then details the formulation of the objective functions before presenting a pose optimisation nearest neighbour algorithm. Section III presents experimental results using data collected both in our laboratory and on-site, comparing the new approach to alternative approaches [5], [7], [11]. Section IV discusses the limitations and possible drawbacks to the approach. Section V provides conclusions and future work.

The authors are with the Centre for Autonomous Systems at the University of Technology, Sydney. E-mail: Gavin.Paul@uts.edu.au



(a) (b)

Fig. 1. *a*) A 7DOF biologically climbing robot with 6 toes and depth and RGB cameras mounted in the "hand", *b*) A climbing robot walking along the wall of a steel bridge box girder tunnel environment.

II. Methodology

A. Robot and Sensor Model

Consider a robot, such as the bio-inspired climbing robot shown in Fig 1*a*, positioned at a base location described by a homogeneous transform, ${}^0T_b$. The robot can be described as an n-dimensional kinematic chain such that given the joint angles, $\mathbf{q} = [q_1, \dots q_n]^T$, the end-effector location is,

$$ {}^bT_f(\mathbf{q}) = \prod_{i=1}^{n} {}^{i-1}T_i(q_i) \tag{1} $$

Where a depth camera is rigidly mounted on the end-effector, and its position relative to the end-effector is given by ${}^fT_c$, then the position and orientation of the sensor is,

$$ {}^0T_c(\mathbf{q}) = {}^0T_b\,{}^0T_f(\mathbf{q})\,{}^fT_c, \tag{2} $$

which describes both the cameras center position, $\mathbf{p_c}(\mathbf{q})$ and a projection line from the camera's center normal to the image plane, $\mathbf{n_c}(\mathbf{q})$.

A depth camera, such as a Structure Sensor, returns a grayscale image with resolution $M_d \times N_d$ (e.g. 640 x 480) of depth values, $D = d_{m,n} \forall m \in \{1, ..., M_d\}, n \in \{1, ..., N_d\}$. By using the camera's intrinsic parameters from calibration and trigonometry, each pixel of the depth image can be turned into a point cloud, $P = p_{m,n} \forall m \in M_d, n \in N_d$. Since the point cloud returned is an "organised" point cloud, it is also possible to rapidly compute a set of normals using the "Average 3D Gradient" technique of Integral Image Normal Estimation [12] inside the Point Cloud Library.

B. Plane Set and Feature Detection

In a repetitive structural tunnel such as Fig 1*b*, the nature and key features of the environment are known *a priori*, in that it is bounded by four sets of main coplanar plane patches, or manholes plates (perpendicular to the tunnel direction), as well as many smaller plates with rivets connecting these.

When looking down a tunnel such that the camera's viewing direction is almost parallel to most surfaces other than the manhole plates, the points on the surfaces are generally noisy and patchy (due to spurious sensor readings, and to the nature of a depth camera that projects light and requires a reflection from the surface). Extracting planes largely eliminates sensor noise and turns the list of 3D points, $\{p_i\}_{i=1,\dots N_d \times M_d}$, into a set, $\mathbf{\Pi}$, of N_Π planes, $\Pi_i = \{\mathbf{p_{\Pi,i}}, \mathbf{n_{\Pi,i}}\}$ for $i = 1, \dots N_\Pi$, where each plane consists of a point, $\mathbf{p_{\Pi,i}}$ that is within that ith plane's region, and a normal, $\mathbf{n_{\Pi,i}}$ of the ith plane.

Our plane growing algorithm [6], based on [13] and [14], uses an "organised" point cloud with associated normals. Seed points are selected and tested against neighbouring points to try and combine them into a larger plane group and update the plane model. Points are iterated through using the test/add step until no point can be added, then a new plane is grown. A fast normal estimation algorithm [12] is used to compute surface normals and update the plane model.

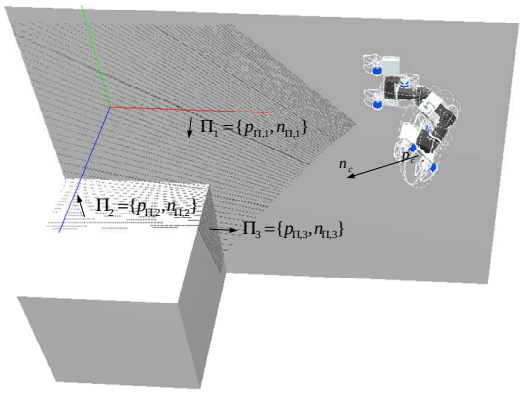


Fig. 2. Robot in a pose taking a scan of a surface with a cube on it so 3 planes are visible from the sensor. The plane equations, along with the camera position and normal, are annotated.

Environment map templates can be described more compactly with plane sets compared to point clouds as shown in Fig. 2. Given a set of planes, a map of the environment can be generated using a combination of prior knowledge and template matching to detect the key surface features that are within expected bounds. In the case of a robot inside a tunnel environment it is necessary to detect the surrounding tunnel. Template-based manhole plate or tunnel detection [6] has been shown to robustly detect and generate an environment map based upon prior knowledge of the context of an application, provided that at least two sets of parallel walls can be detected simultaneously.

The Surface Focused Nearest Neighbour (SFNN) approach shown in Fig. 3 is provided with the latest plane set. SFNN determines the nearest neighbour poses, computes the values of objective functions (devised based upon key features), it then checks pose safety, sorts viewpoints and moves the robot to the best viewpoint to take a scan, which is used to update the map and provide a plane set for the next iteration.

C. Objective Functions

In order to make decisions about where the robot should move next so as to detect surface features, a type of "pose selection" [15] is required to find a pose, $\mathbf{q}$, which corresponds to a desirable viewpoint. A relationship between $\mathbf{q}$ and the quality of the resulting viewpoint is thus established.

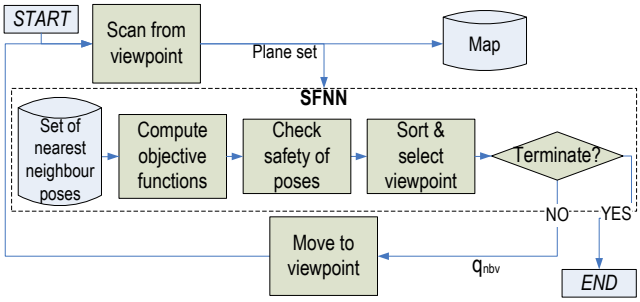
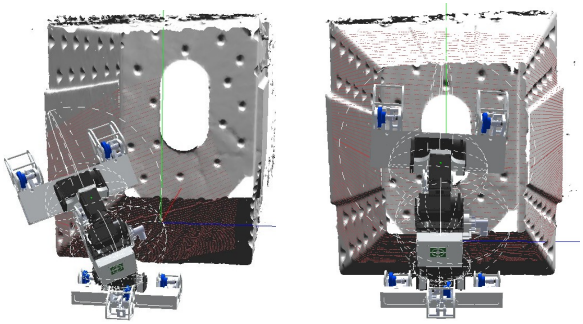


Fig. 3. Surface Focused Nearest Neighbour (SFNN) approach overview.

As shown in (2), it is possible to compute the position and orientation of the sensor tool, ${}^0T_c(\mathbf{q})$. For the potential set of key feature objectives that encode the efficacy of a view of a tunnel situation, Fig. 4*a* shows an example of a scan from a low scoring and undesirable pose, while Fig. 4*b* shows the robot pose and scan from a high-scoring and thus more desirable pose where key features are likely to be detected.



(a) (b)

Fig. 4. For the given set of key feature objectives these are examples of: *a*) Low-scoring undesirable pose, *b*) High-scoring desirable pose.

An optimisation approach such as AXBAM [7], where a sample of the solution space is selected and the contending viewpoints are ranked, is unsuitable. The high dimensionality of the problem makes it intractable to sample the solution space finely enough to ensure an appropriate sensing viewpoint. Instead, the viewpoint determination process requires a fine-tuning optimisation approach. Therefore each objective function will be formulated as a "score function", $g_i(\mathbf{q})$, which reflects whether key identifying features of the environment have been observed, or are likely to be observed.

Given a candidate set of nearest neighbourhood configurations, the highest scoring candidate is selected. The objectives will be shown for the Tunnel and Manhole Plate cases. It would be straightforward to replace these objectives with alternative surface or key feature detection objectives in order to use the nearest neighbourhood pose search to explore for different surface feature or templates.

1) Tunnel Alignment Case: For each of the N_Π planes in a plane set, given the ith plane normal, $\mathbf{n_{\Pi,i}}$ and plane location, $\mathbf{p_{\Pi,i}}$, and the candidate robot pose, $\mathbf{q}$ that results in the camera being located with a position, $\mathbf{p_c}(\mathbf{q})$, and normal, $\mathbf{n_c}(\mathbf{q})$, a score is computed. For the ith plane, the sensor's centre ray meets an object's surface at an angle of incidence. This angle is defined as the absolute value of the dot product between the camera normal and the plane normal,

$$\phi_i(\mathbf{q}) = |\mathbf{n_c}(\mathbf{q}) \cdot \mathbf{n_{\Pi,i}}| \tag{3}$$

which is maximised (i.e. unity) when the camera is looking directly at a plane such that the angle between the plane normal and the camera viewing direction is $0°$ (or $180°$), and a value of zero occurs when the angle between the camera viewing direction and the plane normal is $90°$.

A plane with a large area is more trustworthy and less prone to sensor noise, therefore a plane's score is scaled by its area. So if there are two planes Π_i,Π_j, with equal $\phi_i(\mathbf{q})$, and if Π_i has a larger area (e.g. $0.5m^2$), then it will have a higher score than Π_j, which has a smaller area (e.g. $0.25m^2$). The plane's area, $a_{\Pi,i}$ is computed using a technique in [6].

The sensor to plane distance is also considered since short distance readings are more trustworthy and sensor errors are magnified by distance. So where $||x||$ is the length of the resulting vector, the distance is calculated as,

$$d_1(\mathbf{p_c}(\mathbf{q}), \mathbf{p_{\Pi,i}}) = ||\mathbf{p_c}(\mathbf{q}) - \mathbf{p_{\Pi,i}}|| \tag{4}$$

The score for a single plane, Π_i and pose, $\mathbf{q}$ is given by,

$$s_t(\Pi_i, \mathbf{q}) = \left| a_{\Pi,i} \frac{1 - \phi_i(\mathbf{q})}{d_1(\mathbf{p_c}(\mathbf{q}), \mathbf{p_{\Pi,i}})} \right| \tag{5}$$

In the tunnel alignment case, the objective function for the plane set is made by combining the scores for each plane,

$$g_t(\mathbf{q}, \mathbf{\Pi}) = \sum_{i=1}^{N_\Pi} s_t(\Pi_i, \mathbf{q}) \tag{6}$$

2) Manhole Alignment Case: In the case where the RGB-D image is analysed to find a particular ellipsoidal feature which appears to be a manhole plate, then the objective function is different. This score takes into account the area of the plane, angle to the plane, the distance to the plane, and in certain cases where previous context exists about the existence of the manhole, an estimate of the distance to the manhole's center. This objective function incorporates three sub-scores and several filters to remove certain planes, such as those that are almost parallel to the camera view (i.e. $\phi_i(\mathbf{q})$ is larger than θ_{ignore}) radians, or are too far away, i.e. further than a threshold, d_{max}, which is heuristically determined to be beyond the required level of accuracy for the camera.

The planes that remain after filtering are processed. The angle between the camera and the plane normal varies from $0°$ to $90°$, and hence the dot product varies from 1 to 0 (i.e. desirable to undesirable). An example of the two extreme cases is: (a) the ideal case, when the angle is 0 and the camera is looking straight at the planes which will produce a high score of 1; and (b) non-ideal case, when the angle is less than $45°$ and hence will produce a low score. The angle sub-score is calculated as,

$$s^*_{angle,i} = \left| \frac{cos^{-1}(||n_c \cdot P_n||) - \pi/4}{\pi/4} \right| \tag{7}$$

Planes that are further from the camera are more prone to spurious results, and hence are made to decrease the plane score, whereas a small distance will leave the score unaffected. In the case of two planes with equal normal and area but Π_i is 1m away the Π_j is 2m away, then Π_i should have a higher score than Π_j. The distance from the camera image plane to the center of the plane is calculated as,

$$d_2(\mathbf{p_c}(\mathbf{q}), \mathbf{p_{\Pi,i}}) = 1 - (\mathbf{n_{\Pi,i}} \cdot (\mathbf{p_c}(\mathbf{q}) - \mathbf{p_{\Pi,i}}) - d^*) \quad (8)$$

which is different from the point-to-point distance (4), and includes an ideal distance, d^* where detection of the key features is most straightforward and thus positively affects the score. To ensure the distance sub-score is with the range 0 to 1, it is bound using,

$$s^*_{dist,i} = argmax(0, 1 - |d_2(\mathbf{p_c}(\mathbf{q}), \mathbf{p_{\Pi,i}}) - d^*|) \quad (9)$$

which is largest (i.e. 1) when $d_2(\mathbf{p_c}(\mathbf{q}), \mathbf{p_{\Pi,i}}) = d^*$, and smallest (i.e. 0) when the distance is more than 1m from d^*.

If there have been previous scans that have detected a manhole center, $\mathbf{p_{mc}}$ but have failed the confidence test due to suspected inaccuracies, then the estimated $\mathbf{p_{mc}}$ is used to improve the sub-score for poses that position the center ray of the camera at the manhole center as follows,

$$s^*_{center,i} = 1 - ||\mathbf{p_c}(\mathbf{q}) - \mathbf{p_{mc}}|| \quad (10)$$

where $||x||$ denotes a vector's length. If there is no *a priori* context then $s^*_{center,i} = 0$ and does not affect the score.

The plane's score is based upon the summation of three sub-scores, scaled by the area of the plane,

$$s_m(\Pi_i, \mathbf{q}) = a_{\Pi,i}(s^*_{angle,i} + s^*_{dist,i} + s^*_{center,i}) \quad (11)$$

so a plane with a small area multiplied by a poor angle and with a low distance score will receive an overall low score, while a plane with a larger area that is directly in front of the camera, at a good distance will receive a high score.

In the manhole alignment case the objective function for the plane set is made by combining the scores for each plane,

$$g_m(\mathbf{q}, \mathbf{\Pi}) = \sum_{i=1}^{N_\Pi} s_m(\Pi_i, \mathbf{q}) \quad (12)$$

D. Nearest Neighbour Optimisation

The SFNN algorithm is designed towards the rapid discovery of a valid joint configuration for a viewpoint of a target which contains significant key feature information. A joint vector, $\mathbf{q}$, is determined to achieve a viewpoint which will maximise the score from the objectives and thus enable the targeted surface features to be appropriately sensed and a template map generated.

A viewpoint is the result of the robot being in a particular joint configuration, $\mathbf{q}$, which must fall within the physical angular limitations. For the $i \in \{1, \dots n\}$ joints these are defined as the maximums, $\mathbf{q}_{i,max}$, and minimums, $\mathbf{q}_{i,min}$. In exploration, the solution space, $\mathbf{Q}$ is sampled so all solutions that fall within the joint limits are allowable, and those that fall outside the space are discarded.

Algorithm 1: Nearest Neighbour Pose Selection

Input: $\mathbf{q}_{curr} \longleftarrow$Current robot pose , $\mathbf{\Pi} \longleftarrow$Set of planes, $\mathbf{Q} \longleftarrow$Set of poses under consideration
Output: $\mathbf{q}_{nbv}, \mathbf{Q}$
$\mathbf{Q}_n$ = Generate_Neighbour_Poses($\mathbf{q}_{curr}, M$);
$\mathbf{q}_{nbv} = \emptyset$;
best_neighbour_score$= 0$;
if $\mathbf{Q}_n \neq \emptyset$ **then**
  for $\mathbf{q} \in \mathbf{Q}_n$ **do**
    $score = g_i(\mathbf{q}, \mathbf{\Pi})$;
    if $score >$ *best_neighbour_score* **then**
      best_neighbour_score $= score$;
      $\mathbf{q}_{nbv} = \mathbf{q}$;

The set of nearest neighbour poses can be generated using Algorithm 1 and then by comparing the scores and selecting the pose associated with the best score from amongst the candidates. Beginning with the current pose, $\mathbf{q}_{curr} = (1, \dots, j)$, a vector of j joint angles, each joint angle in $\mathbf{q}_{curr}$ is iterated over, adding or subtracting a chosen angle δq, such that if pose $\mathbf{q}_t$ at time, t is safe, given a map, M, and if pose $\mathbf{q}_t + \delta q_i$ is safe, then the trajectory, $\mathbf{q}_t + v \times \delta q_i, \forall v \in \mathbb{R}, 0 \leq v \leq 1$ will also be safe and no extra trajectory path planning is required. The resulting pairs of poses are then added to $\mathbf{Q}_n$.

In order to ensure the safety of the robot during the selection of candidate poses, collision avoidance is implemented using the ellipsoidal bounding fields around each robot link based upon [4]. The ith joint is enclosed in ellipsoidal virtual bounding fields, centred at $\mathbf{p}_{c,i}$ and with semi-principal axes *eccentricity parameters*, $[a_{e,i}, b_{e,i}, c_{e,i}]$. These ellipsoids are used for collision checks. An obstacle (or unexplored voxel), $\mathbf{p} \in \mathsf{P}$, within an ellipsoid has an algebraic distance less than 1, therefore all known obstacle, and unknown voxels are checked for each joint to ensure that none of the them are inside any of the ellipsoids. For each of the joints, q_i in $\mathbf{q}$, the corresponding ellipsoid's algebraic distance, $dist(\mathbf{q})$ to all obstacles and unknown voxels is returned using

$$dist(\mathbf{q}) = \min_{\mathbf{p} \in \mathsf{P}} \left(\min_{i \in \{1, \dots n\}} \left(d^T \cdot \begin{bmatrix} a_{e,i}^{-2} & 0 & 0 \\ 0 & b_{e,i}^{-2} & 0 \\ 0 & 0 & c_{e,i}^{-2} \end{bmatrix} \cdot d \right) \right) \quad (13)$$

where $d = \left(\mathbf{p}^{\,0}T_i(\mathbf{q})^{-1} - \mathbf{p}_{c,i} \right)$. Provided that the smallest $dist(\mathbf{q})$ from all ellipsoids to all obstacle and unknown points in the environment is greater than 1 then the points lie outside all ellipsoids and the pose is safe.

Practically, the joints of any robot cannot be moved precisely to infinitesimally small increments. Therefore, in order to prevent against a potential infinite loop condition where the same poses continue to be identified as the next best potential pose, poses that are used are stored rounded to the nearest integer. Each time new poses are generated, their rounded integer pose is checked against the stored database

for the current base location. Poses are discarded if their rounded value has been visited. In the case where all nearest neighbour poses have been determined as being visited, or the number of exploration poses goes over a pre-specified application-specific threshold, then exploration is terminated.

III. Results

Two experiments have been conducted in replicas of steel bridge tunnel environments shown in Fig. 5 using a 7DOF climbing robot with two cameras mounted to the end-effector: a Structure Sensor depth camera, and a Logitech C930e RGB camera. The robot can attach one or two footpads to steel surfaces in the environment using the 3 actuatable, permanent-magnet toes in both footpads.

Experiment 1 was conducted with a simulated climbing robot in two environments: (a) simple tunnel consisting of 4 parallel walls connected together with L-beams (Fig. 5*a*); (b) a tunnel environment collected on-site in a real-world steel bridge box girder containing 12 approximately parallel planes, rivets, rust and a manhole plate (Fig. 5*b*). The robot is always attached to one surface, simulated depth data is collected by ray casting into the environment and generating a mock image without noise that replicates the ideal scan.

Each RGB-D frame is collected and processed: the resultant point cloud data is triangulated to generate a mesh and plane set which is analysed to detect the key features (e.g. the tunnel and manhole plate) [6]. The dimensions and length of the tunnel are initially unknown to the detection process. If the detection process fails to output the tunnel dimensions then exploration continues, or if valid dimensions are determined then exploration terminates.

In order to compare the presented SFNN approach, several other alternative approaches have also been implemented and are described in Table I along with their advantages and disadvantages.

Method	Advantages	Disadvantages
SFNN: Surface Focused Nearest Neighbour approach presented in this paper	Deterministic, can detect key features, computationally inexpensive	Objective functions must be formulated
RAND: selects a pose at random from the nearest neighbour candidates instead of using objective functions	Least complexity, no objective functions, simple to implement	Stochastic, unpredictable, expected to regularly fail, cannot guarantee map improvement
NN: Based on [5] it estimates the information gain (i.e. unknown voxel penetration count) for each scan	Deterministic, improves surface & occupancy maps, generally faster than AXBAM [7]	Unlikely to detect key features, information gain computation is expensive
NNB: based upon [11] as an extension to NN with the added facility to backtrack if no good option is found	Deterministic, improves surface & occupancy maps, more likely to succeed than NN	Unlikely to detect key features, similar computational overhead to NN

TABLE I

Description and characteristics of comparative methods.

In Experiment 1, the robot is initialised at a different base location for each environment. At each base location there are 20 initial poses in which the robot starts for all approaches. Then the different approaches are used to select the next best viewpoint, the simulated robot moves there and takes a depth image. The image is analysed by a detection module, and the tunnel or manhole is either found or it isn't. Once the tunnel or manhole is found, the test is stopped and the number of scans required is recorded. If no desired feature is found within $c_{max} = 20$ iterations then the test is terminated and c_{max} is recorded. The mean iteration count required, $\bar{c_1}$ is then computed as well as the standard deviation (SD), (σ_1) and shown on the left in Table II.

	Experiment 1		Experiment 2	
Map:	*a*	*b*	*c*	*d*
mean(SD)	$\bar{c_1}(\sigma_1)$	$\bar{c_1}(\sigma_1)$	$\bar{c_2}(\sigma_2)$	$\bar{c_2}(\sigma_2)$
SFNN	7.1(7.1)	4.3(2.4)	2.9(2.3)	5.3(4.1)
RAND	9.8(7.1)	8.5(5.97)	N/A	N/A
NN	9.7(8.7)	15.3(8.4)	N/A	N/A
NNB	9.7(8.7)	6.2(3.7)	N/A	N/A

TABLE II

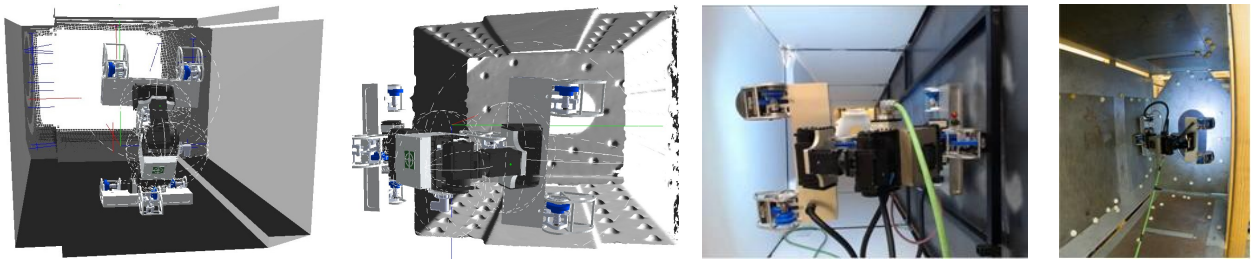
Experimental Results.

Experiment 2 was conducted in two lab scenarios. The first as shown in Fig. 5*c* is a robot attached to the side wall and scanning into the simple lab tunnel which consists of 4 parallel walls. The second as shown in Fig. 5*d* is a replica bridge tunnel and manhole including rivets. The robot walked to a position where it is close enough to the manhole to detect the key environment features (i.e. the tunnel or manhole).

For Experiment 2, one base location is used for each environment with the centre toe facing approximately into the tunnel. Our approach is run from 10 different initial poses and the number of scans required to find a solution is once again recorded. Note that the maximum number of iterations allowed is set to 20, due to time constraints and to limit the wear on the robot's joints. The right side of Table II shows the mean scan count, $\bar{c_2}$ and standard deviation, (σ_2) calculated for the 10 poses used for each environment map.

IV. Discussion

It has been shown that the proposed SFNN approach works well for the target environments and rapidly converges upon poses that allow the key features of an environment to be observed. The previous approaches (NN, NNB and thus by extension, AXBAM) do not perform much better than simply randomly selecting the next pose, especially in environment *a* which has the fewest features. The proposed approach is able to determine the structure rapidly, particularly in the case of feature-rich environments such as *b* that is based upon field scan data, and the two real-world cases (i.e. *c* and *d*). An additional benefit is that the path and pose is found as a by-product of the search. This path could be improved upon using a separate planner, however the results are already safe and any additional planning is not likely to reduce the movement time significantly. The tunnel and manhole detection is still by far the most time consuming part with each frame taking several seconds to process, while



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

Fig. 5. Simulated environment and robot: *a*) Tunnel only based on field measurements; *b*) Tunnel and manhole based on field scan data. Real environments and climbing robot: *c*) simple laboratory tunnel; *d*) Replica field tunnel with rivets and nearby manhole.

the proposed SFNN approach takes between 400-600ms to compute the objective function results, optimise and confirm the safety of the path to the next pose.

If the initial pose is looking directly at a side wall or roof, then it is theoretically possible that the optimisation may not converge and may stop or get caught in a local minima as has been observed in some difficult simulated cases. However this behaviour was only observed in 5% of cases, and only in the simulation environment with the artificially decreased feature set (i.e. *a*) where alternative approaches also found no solutions. When using real-world data (i.e. *b-d*) the key environment features were successfully found in every instance, and thus for SFNN the upper limit to force the optimisation to terminate was never reached.

V. Conclusions

This paper has presented an exploration approach for robots to determine future sensing actions that can best facilitate the building of surface maps within repetitive structured environments. Key features that describe an environment are used to create objective functions. Using these objective functions, the approach repeatedly searches through a sequence of nearest neighbour poses in configuration space from which the next observation will be made, until the key features are observed. The approach in this paper has been experimentally evaluated using a climbing robot that must adhere to surfaces in order to traverse and navigate in an environment. The approach is compared to previous exploration approaches and has been shown to converge upon viewpoint poses, which are predicted to contain sufficient key features that can be observed, given starting poses that are sufficiently close.

Future work will evaluate improvements and additions to the objective functions, such as the maximisation of depth values in depth images to aid in quickly finding a viewpoint that observes down the length of the tunnel, as well as increasing the number of objective functions that can be simultaneously used in this exploration approach.

Acknowledgments

This work is supported by the NSW Roads and Maritime Services, and the Centre for Autonomous Systems (CAS) at the University of Technology, Sydney

References

[1] P. Ward, G. Paul, P. Quin, D. Pagano, C. Yang, D. Liu, K. Waldron, G. Dissanayake, P. Brooks, P. Mann, W. Kaluarachchi, M. P., and L. Matkovic, "Climbing robot for steel bridge inspection: Design challenges," in *9th Austroads Bridge Conference*, Sydney, 2014.

[2] M. Eich and T. Vogele, "Design and control of a lightweight magnetic climbing robot for vessel inspection," in *19th Mediterranean Conf. on Control Automation*, Corfu, 2011, pp. 1200–1205.

[3] 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.

[4] 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. 7-8, pp. 543–554, 2011.

[5] P. Quin, G. Paul, A. Alempijevic, D. Liu, and G. Dissanayake, "Efficient neighbourhood-based information gain approach for exploration of complex 3d environments," in *IEEE Int. Conf on Robotics and Automation (ICRA)*, 2013, pp. 1343–1348.

[6] 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 *IEEE Int. Conf. on CYBER Technology in Automation, Control, and Intelligent Systems*, 2015, pp. 1097–1102.

[7] G. Paul, S. Mao, L. Liu, and R. Xiong, "Mapping repetitive structural tunnel environments for a biologically inspired climbing robot," in *18th Int. Conf. on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, 2015, pp. 325–333.

[8] S. Son, S. Kim, and K. Lee, "Path planning of multi-patched freeform surfaces for laser scanning," *The International Journal of Advanced Manufacturing Technology*, vol. 22, no. 5-6, pp. 424–435, 2003.

[9] J. Wang, D. Gu, Z. Yu, C. Tan, and L. Zhou, "A framework for 3d model reconstruction in reverse engineering," *Computers & Industrial Engineering*, vol. 63, no. 4, pp. 1189–1200, 2012.

[10] S. Sehestedt, G. Paul, D. Rushton-Smith, and D. Liu, "Prior-knowledge assisted fast 3d map building of structured environments for steel bridge maintenance," in *IEEE Int Conf. on Automation Science and Engineering (CASE)*, 2013, pp. 1040–1046.

[11] P. Quin, G. Paul, A. Alempijevic, and D. Liu, "Nearest neighbour exploration with backtracking for robotic exploration of complex 3d environments," in *Australasian Conf. on Robotics and Automation*, 2013, pp. 1343–1348.

[12] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab, "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 2684–2689.

[13] J. Xiao, J. Zhang, B. Adler, H. Zhang, and J. Zhang, "Three-dimensional point cloud plane segmentation in both structured and unstructured environments," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1641–1652, 2013.

[14] C. Feng, Y. Taguchi, and V. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 6218–6225.

[15] G. Paul, N. Kirchner, D. K. Liu, and G. Dissanayake, "An effective exploration approach to simultaneous mapping and surface material-type identification of complex 3d environments," *Journal of Field Robotics (S.I. 3D Mapping)*, vol. 26, no. 11-12 SI, pp. 915–933, 2009.