Efficient Environment Guided Approach for Exploration of Complex Environments

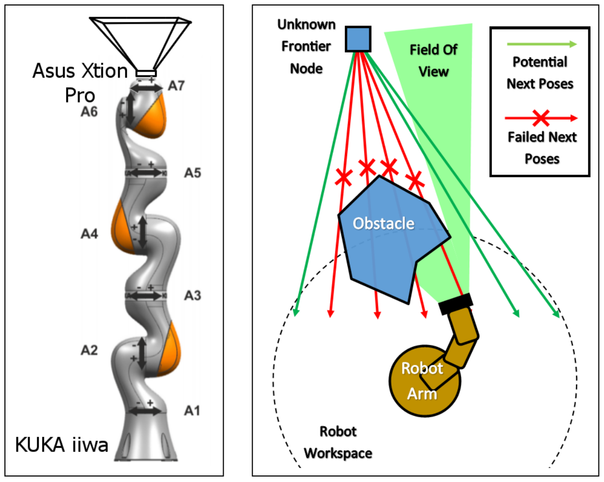
Daniel Butters[1], Emil T. Jonasson[2], Robert Stuart-Smith[1] and Vijay M. Pawar[1,3]

***Abstract*— Remote inspection of a complex environment is a difficult, time consuming task for human operators to perform. The need to manually avoid obstacles whilst considering other performance factors i.e. time taken, joint effort and information gained represents significant challenges to continuous operation. This paper proposes an autonomous robotic solution for exploration of an unknown, complex environment using a high DoF robot arm with an eye in hand depth sensor. The main contribution of this work is a new strategy to find the next best view by evaluating frontier regions of the map to maximise coverage, in contrast to many current approaches which densely sample joint or workspace configurations of the robot. Multiple utility functions were evaluated that showed different behaviours. Our results indicated that the presented algorithm can explore an arbitrary environment efficiently while optimising various performance criteria based on the utility function chosen, application constraints and the desires of the user.**

I. INTRODUCTION

For many situations, the task of teleoperating a robot to inspect an unknown environment is either too difficult or impractical to work effectively, particularly for continuous monitoring. Whilst there are a number of real-world scenarios that would benefit from robust autonomous inspection, the scale and complexity of these applications poses great challenges for established methods of navigation and data capture [1].

Inspection subjects can range from search and rescue to industrial infrastructure, such as a large network of tunnels or a cluttered nuclear site. Monitoring these environments presently rely on solutions that involve a human operator. In situations that are time-critical, these methods often lead to areas of missing information, and sub-optimal path planning [2]. When monitoring the health of large scale sites, this can become a limiting factor, for example when using a digital twin asset management tool [1] in being able to anticipate faults and make better decisions that ensure operational efficiency. By using autonomous robots with collision aware planning strategies, in this paper we explore solutions that help maximise mapping coverage whilst also providing robust solutions to the challenges of operating in a cluttered environment. We believe these methods will enable further research in exploring complex spaces to produce more complete data sets for better inspection performance.



(a) 7 DoF Robot arm and sensor. (b) Concept diagram of the algorithm from a top-down view.

Fig. 1. The hardware used and concept diagram. Sampling the robot workspace for sensor positions reduces the solution space from 7 to 3 dimensional space.

For a robot to autonomously explore an unknown environment, it must be able to identify new locations to move to and acquire new information. As discussed by Yamauchi et al [3], the over-arching question can be described as "Given what you know about the world, where should you move to, to gain as much new information as possible?" Introduced as frontier-based exploration, their work investigates strategies that help robots move between known free-space and unknown space explore the environment. To date, researchers are beginning to evaluate the effectiveness of these strategies on real robots. Further, building upon work by Shade et al [4], in order to develop generalised solutions beyond specific operating conditions, there is a clear need for methods that enable robots to: 1) explore efficiently in three dimensions 2) require no prior knowledge of the environment, 3) operate without infrastructure which might provide pose estimation (such as fixed camera networks, radio beacons, or pre-placed markers) 4) be scalable to larger environments.

In this paper, we present a novel approach to collision aware frontier exploration and demonstrate it working on a real 7 DoF (degree of freedom) robot. To test the performance of our method, we developed an evaluation framework using a set of complex scenarios, specifically comparable to environments observed when exploring a nuclear site. Based upon these results, we highlight the trade-offs in robot behaviour depending on different priorities of the exploration

[1]Authors are members of the Autonomous Manufacturing Laboratory within the Department of Computer Science, University College London, Gower Street, WC1E 6BT, UK. daniel.butters.16@ucl.ac.uk

[2]Author works at Remote Applications in Challenging Environments (RACE), UK Atomic Energy Authority, Culham Science Centre, Abingdon, OX14 3BD, UK. emil.jonasson@ukaea.uk

[3] Corresponding Author: v.pawar@ucl.ac.uk

strategy employed. In doing so, we extend previous work in developing autonomous exploration strategies deployed on high-DoF robot arms. We also further our understanding for how frontier-based approaches could be used to automate the monitoring of large scale and complex environments without human intervention.

II. RELATED WORK

The topic of motion planning for autonomous exploration has been well studied in recent years. Blaer et al [5] described an approach for data acquisition in a large environment using 2D next best view planning. For systems with high degrees of freedom in planning, strategies for exploration are commonly greedy methods that iteratively determine the next best view. Yamauchi et al [3] demonstrated the first instance of frontier based exploration, where this was described as the boundary between observed and unknown regions.

Vasquez-Gomez et al [6] proposed a method for using octrees to represent the environment being explored, and for performing fast visibility computations using hierarchical raycasting. This was only applied in an environment mostly free of occlusions or obstacles. Shade et al [4] used frontier based exploration with a potential vector field to enable exploration of the environment, but encountered high computational costs with this method.

Many current approaches evaluate a large number of robot configurations for their potential information gain. For example, Paul et al [7] described a method to evaluate the information gain of a viewpoint by raycasting densely through the field of view of the sensor. With high DoF systems, methods that require sampling these configurations can become computationally expensive. Quin et al [8] extends this approach by optimising the coverage of the path between viewpoints. However, this was demonstrated in environments with limited complex objects or obstacles within the robot workspace.

The metric used to determine the next best view can vary between different approaches. An algorithm which used utility functions for choosing next best views with a high DoF arm was proposed by Vasquez-Gomez et al [9], demonstrated in an environment with no obstacles. An approach to surface exploration by continually optimising the coverage of the surface by an MAV in motion was proposed by Song et al [10]. This was demonstrated in simulated environments where obstacles were sparse. An approach which combined space exploration and object recognition tasks by using different metrics for each objective was proposed by Friedrich et al [11], however this was operated over a small area with very simple obstacles and a low DoF robot arm.

The application of exploration in hazardous, particularly nuclear environments was explored by Nagatani et al [12], however this was a teleoperated robot.

Compared to the state of the art, this paper demonstrates a system that uses frontier based exploration with adaptively scaled raycasting to quickly determine the visibility of unknown areas of the map. The raycasting was performed from frontier voxels to the robot workspace to determine visibility given the constraints of the robot. The next best view was then chosen by ranking the visible voxels using one of several utility functions. Unlike other studies to our knowledge, this algorithm was tested in environments with dense obstacles and many occlusions in order to show how the algorithm is robust to complex environments which are classically difficult to explore autonomously.

III. ENVIRONMENT GUIDED EXPLORATION ALGORITHM (EGEA)

This paper proposes a method for exploring a volume with a fixed base robotic arm using an eye in hand depth sensor. The volume to be explored, V, is predefined with a bounding box which is fixed with respect to the world frame. In this study the shape of the volume was a cuboid which contained the environment being explored.

This algorithm iteratively searches for the next best motion plan for the robot arm, given the current information known about the environment. The motion plan is chosen via various utility functions, described in the following section.

The proposed method directly estimates the potential information gain from different regions of the map by measuring the number density of unknown voxels on the frontier of the known regions of the map. Then, raycasts are performed to evaluate which regions on the frontier are possible to be viewed by the robot. EGEA evaluates regions of the map directly for their information gain and then a path plan is found to view that area. This approach aims to significantly reduce the number of raycast operations performed when compared to other methods because the robot workspace is sampled over 3 dimensions for sensor positions instead of a high dimensional sampling of robot joint configurations (i.e. a 7 DoF robot results in a 7 dimensional problem).

Figure 1(b) shows a concept diagram of the algorithm from a top down view. This algorithm uses the Octomap library [13] to record current knowledge of the environment. Octomap is a 3D voxel grid which allows efficient neighbour searching and raycasting. Typically, each voxel can be labelled as one of three states: Occupied, Free and Unknown. In this work the voxels representing the environment include a fourth state: Unreachable. A voxel can only be one state at a time.

Occupied - The voxel has been observed as an obstacle.

Free - The voxel has been observed as free of obstacles.

Unknown - The voxel has not been observed.

Unreachable - The voxel can not be observed by the robot. The voxel is obscured by other Occupied voxels, or the path planner has failed to find a path to observe the voxel.

Algorithm 1 provides pseudocode for EGEA, starting with the robot in a random pose within an unexplored environment. The variables are defined in Table I.

At the first iteration, the robot is at a random pose and it is assumed that the sensor can observe at least one voxel within the bounding box of the environment. The robot is also assumed to have enough free space to make an initial movement to view the environment, and then move into

Algorithm 1: Pseudocode for EGEA

```
Input : Octomap $M$, Octomap $RoughMap$,
        Frontier Voxels $F$
while $While\ number\_Of\_Voxels(F) > 0$ do
    $F = M.get\_Frontier\_Voxels()$;
    $L_{rough} = \lceil \sqrt[3]{number\_Of\_Voxels(F)} \rceil \times L_M$;
    $RoughMap.set\_Voxel\_Length(L_{rough})$;
    $RoughMap.calculate\_IG(F)$;
    $TargVoxels =$ set of Voxels in $RoughMap$ where
      $IG(Voxel) > (AveIG - 1)$;
    if $Method\ 1$ then
        $TargVoxel = \max_{(IG)}(TargVoxels)$;
        $P_{motion} = find\_Motion\_Plan(TargVoxel)$;
    else if $Method\ 2$ then
        $TargPlans = find\_Motion\_Plan(TargVoxels)$;
        $P_{motion} = \max_{(\frac{IG}{jointEffort})}(TargPlans)$;
    else if $Method\ 3$ then
        $TargPlans = find\_Motion\_Plan(TargVoxels)$;
        $P_{motion} = \min_{(jointEffort)}(TargPlans)$;
    $execute(P_{motion})$;
end
```

regions of the workspace containing obstacles. For every iteration, the Octomap representing the environment, M, is inputted. The set of frontier voxels, F, is generated by selecting all Unknown voxels that are adjacent to Free or Occupied voxels. Rays are cast between each voxel in F and sampled positions within the robot workspace. Rays are unsuccessful if they intersect with an Occupied voxel. If no rays successfully reach the robot workspace, the voxel state is set to Unreachable and removed from F.

RoughMap, described in Table I, counts the frontier voxels that intersect with each voxel in *RoughMap*. This is defined as the Information Gain (IG) of the voxel, and approximates the volumetric number density of frontier voxels within M.

The side length of the voxels in *RoughMap* is determined by the formula $L_{rough} = \lceil \sqrt[3]{number\_Of\_Voxels(F)} \rceil \times L_M$. This formula means that each voxel in *RoughMap* has at least as much volume as the sum total volume of voxels in F. This guarantees that any one voxel in *RoughMap* cannot be saturated by frontier voxels (i.e. if the number of voxels in F is 10, the volume of a voxel in *RoughMap* will be at least 11.) Changing the side length of the voxels in *RoughMap* during each iteration allows the algorithm to roughly sample the environment when there are many frontier voxels, and finely sample the environment when there are few remaining.

The voxels in *RoughMap* are ranked by their IG. The algorithm then chooses a target location from which to observe more of the environment. In this study, six methods for choosing the target and the associated motion plan were considered and divided into three utility functions, called Method 1, 2 and 3 respectively:

Method 1 is the simplest utility function as it does not consider the joint effort of viewing a given target. The voxel with the greatest information gain is chosen as the target. This method was the quickest in terms of computation because only one target is considered.

Variable	Definition
M	The Octomap representing the environment. The voxels in M can be Occupied, Free, Unknown or Unreachable.
L_M	The length of one side of a voxel within M (metres)
F	Frontier Voxels - Set of Unknown voxels within M that are adjacent to Free or Occupied voxels
RoughMap	A separate Octomap that is superimposed over the same coordinates as M. This map generally has a higher voxel side length than M, so multiple voxels within M will intersect with each voxel in *RoughMap*. Each voxel in *RoughMap* is assigned an integer value called Information Gain (*IG*).
L_{rough}	The length of one side of a voxel within *RoughMap* (metres)
IG - (Information Gain)	The integer value assigned to a voxel in *RoughMap* that corresponds to the number of voxels in the set F that intersect with that voxel. This metric approximates the volumetric number density of F.
AveIG	The average Information Gain of all voxels within *RoughMap*
P_{motion}	The result of the path planning when given a target voxel.

TABLE I
VARIABLE DEFINITIONS FOR ALGORITHM 1

Methods 2 and 3 both consider multiple voxels within *RoughMap* as potential targets. In order to reduce the number of targets, only the voxels with $IG > AveIG - 1$ were considered. For each target, a robot motion was planned and the joint effort of this plan was computed. The joint effort is defined as the absolute change in joint angle between start and end states in the path plan, summed over all joints. For methods 2 and 3, the motion plan with the best value of the respective utility function is chosen.

Method 2 chooses the motion plan which maximises the value of the fraction $(\frac{IG}{jointEffort})$. This aims to maximise Information Gain while minimising joint effort.

Method 3 chooses the motion plan which minimises the value of the variable (*jointEffort*). Minimising the joint movements of the robot can result in less wear on the robot and hence reduce the hardware maintenance.

For each utility function, two methods for choosing the plan to view a given target were tested:

First Path means that when a target voxel was chosen, the planner tests potential camera poses within the robot workspace to view the target voxel. As soon as a viable path was found, this path was chosen. This variant is represented in Table II by **(F)**. **Least Effort Path** means that given a target voxel, the planner tests a sample of poses within the robot workspace, and returns the path with the least joint effort. This variant is represented in Table II by **(L)**.

IV. EXPERIMENTAL SETUP

To evaluate the approach proposed in this paper, both hardware and simulated experiments were performed. Three experiments were performed on real hardware, and one experiment was in simulation. All environments and the associated Octomap are shown in Figure 2.

The algorithm was run on the same Intel Core i7-6700 CPU for all experiments. The simulation used the Gazebo Robot Simulation software that is packaged with ROS Kinetic. The Gazebo sensor plugin [14] was used to simulate the same sensing capabilities as the hardware. ROS was used for communication between all systems.

The robot used in both lab and simulated environments was a KUKA iiwa 800, a 7 DoF robot arm mounted to a fixed base. The hardware used is shown in Figure 1(a). The RGBD sensor was rigidly mounted to the end effector along the axis of rotation for the final joint of the robot arm. The RGBD sensor used is the Asus Xtion Pro. This sensor has a range of 0.5m to 5m, and can provide coloured point clouds at a rate of 30 fps with a resolution of 640x480 points. The environments used for exploration are described here.

- **Cluttered Lab -** A large section of a lab, with a 4x8x4m sized bounding box and a total volume of 128m^3.
- **Boxes -** A set of boxes arranged to provide difficult geometries, inside a 1x2x1m bounding box. The obstacles in this environment were within the robot workspace, restricting the motion of the robot.
- **Pipes -** A set of pipes that provided many occlusions inside a a 1x2x1m bounding box. Pipes are inside the robot workspace, restricting the motion of the robot.
- **Simulated Industrial -** A CAD model of the pre-conceptual design of the Upper Port of the EU-DEMO reactor, based on the 2018-RM configuration model [15]. A 4.5x4x4m bounding box was used.

The metrics used to evaluate the performance of the algorithm are listed below.

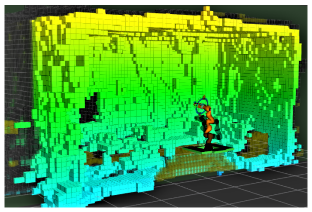
- **Time (mins) -** The total time taken for the algorithm to finish. The algorithm finishes when the Octomap M contains no Unknown voxels, or when the decrease in Unknown voxels between each iteration plateaus.
- **Percent Coverage (%) -** The percentage of the environment that is either Occupied or Free. This was the number of voxels that are Occupied or Free divided by the total number of voxels within the bounding box.
- **Viewpoints Considered -** The total number of potential camera poses that were evaluated by the algorithm when computing the path of the robot.
- **Iterations -** The total number of iterations of the algorithm before stopping.
- **Joint Effort (radians) -** The sum of the absolute rotation carried out by all joints within the robot arm throughout the course of the algorithm.
- **Computation Time (%) -** The percentage of the total time which was spent computing the path of the robot. The remaining portion of time is the robot moving.

For each environment, the algorithm was run 3 times for each of the 6 different utility functions. For all experiments the map was continually updated with sensor information at 30 Hz. The utility functions are described in Section III.

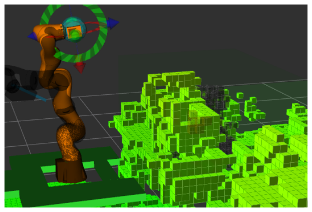
V. EXPERIMENTAL RESULTS

Table II shows the full results for each method and environment. The M1 (F) method has significantly greater joint effort on average than the other methods, and performed the task slightly quicker than other functions. This suggests that a suitable compromise can be found within the three utility functions M2 (F), M3 (F) and M1 (L).

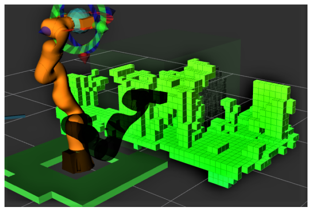
(a) Cluttered Lab Environment (Photo)

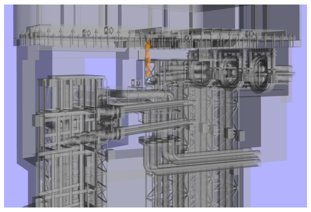

(b) Cluttered Lab Environment (Octomap)

(c) Boxes Environment (Photo)


(d) Boxes Environment (Octomap)

(e) Pipes Environment (Photo)


(f) Pipes Environment (Octomap)


(g) Simulated Industrial Environment (CAD)

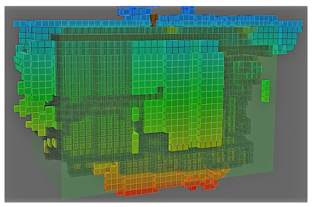

(h) Simulated Industrial Environment (Octomap)

Fig. 2. (Left) Environments used in experiments. (Right) Octomap of each environment after exploration. Occupied voxels are shown as solid colour, Unreachable voxels are black and Frontier voxels are light green and translucent.

Figure 3 suggests that there is an inverse relationship between the time taken to explore an environment and the joint effort required by the system. A possible reason for this relationship is that the two methods that take the most time, M2 (L) and M3 (L) have the two largest average Viewpoints considered (5398 and 4425 respectively). These two methods also have the largest percentage of the time spent computing over all environments at 92.29% and 91.68% respectively. More viewpoints were considered when using these methods, but the robot spent less time and joint effort when moving between poses.

This suggests these two methods are more suitable for applications in which minimising the movement of the robot is a higher priority than the time taken to carry out the exploration. This is particularly relevant in the nuclear industry, where it is important to preserve the components of the robots for as long as possible [16]. When components wear out, repairing the equipment can be expensive and delay other operations being carried out in the environment.

Figure 4 suggests a negative correlation between coverage

	M1 (F)	M1 (L)	M2 (F)	M2 (L)	M3 (F)	M3 (L)
(a) Cluttered Lab Environment						
Time (mins)	**4.33**(1.64)	**6.07**(0.91)	**6.44**(2.71)	**13.67**(4.46)	**5.94**(3.09)	**9.39**(3.81)
Percent Coverage (%)	**86.78**(2.94)	**84.29**(0.99)	**85.69**(0.70)	**80.87**(4.15)	**79.49**(12.01)	**71.85**(17.47)
Viewpoints Considered	**91**(69)	**944**(118)	**2648**(979)	**14809**(4562)	**2451**(1110)	**9853**(4772)
Iterations	**12.33**(4.04)	**16.00**(2.00)	**16.67**(6.03)	**16.33**(4.93)	**16.00**(7.00)	**11.67**(4.93)
Joint Effort (radians)	**104.54**(44.85)	**38.06**(4.70)	**57.54**(5.47)	**24.28**(3.37)	**46.22**(21.93)	**16.30**(7.81)
Computation Time (%)	**79.57**(0.60)	**88.33**(0.99)	**85.97**(1.96)	**95.06**(0.49)	**86.12**(1.83)	**95.03**(0.36)
(b) Boxes Environment						
Time (mins)	**0.75**(0.09)	**1.36**(0.01)	**0.83**(0.26)	**4.84**(0.44)	**1.02**(0.13)	**6.84**(2.03)
Percent Coverage (%)	**94.88**(0.63)	**93.48**(0.91)	**91.72**(1.47)	**94.32**(1.44)	**95.08**(1.18)	**94.31**(1.18)
Viewpoints Considered	**126**(42)	**610**(180)	**569**(182)	**2360**(541)	**583**(99)	**3068**(1152)
Iterations	**9.00**(1.00)	**10.33**(3.06)	**7.33**(1.53)	**7.33**(1.15)	**9.67**(0.58)	**9.33**(2.52)
Joint Effort (radians)	**57.59**(8.51)	**42.24**(11.71)	**33.08**(7.86)	**21.51**(1.75)	**35.10**(4.22)	**22.52**(3.43)
Computation Time (%)	**13.19**(2.05)	**55.54**(8.15)	**40.40**(2.77)	**91.28**(0.56)	**42.23**(5.38)	**91.89**(1.33)
(c) Pipes Environment						
Time (mins)	**0.51**(0.08)	**1.08**(0.20)	**0.63**(0.15)	**3.72**(0.15)	**0.67**(0.08)	**3.96**(0.16)
Percent Coverage (%)	**89.30**(1.06)	**91.49**(5.19)	**90.76**(4.17)	**88.75**(1.98)	**86.02**(2.49)	**89.43**(2.14)
Viewpoints Considered	**102**(53)	**354**(102)	**377**(162)	**1455**(90)	**245**(115)	**1711**(102)
Iterations	**6.33**(1.53)	**6.00**(1.73)	**6.00**(1.00)	**6.00**(1.00)	**5.67**(1.15)	**7.00**(0.00)
Joint Effort (radians)	**37.76**(13.03)	**23.20**(8.07)	**28.05**(8.39)	**17.02**(3.81)	**22.94**(5.10)	**18.79**(1.09)
Computation Time (%)	**12.84**(2.32)	**68.14**(7.46)	**37.56**(1.58)	**91.06**(0.86)	**42.28**(5.42)	**90.15**(0.59)
(d) Simulated Industrial Environment						
Time (mins)	**2.16**(0.47)	**1.22**(0.53)	**3.60**(1.95)	**2.74**(0.77)	**2.03**(0.35)	**2.45**(0.96)
Percent Coverage (%)	**91.33**(4.61)	**59.08**(5.25)	**80.76**(10.15)	**34.92**(11.86)	**79.57**(7.60)	**44.64**(9.78)
Viewpoints Considered	**210**(137)	**767**(257)	**3335**(2159)	**2970**(884)	**1811**(151)	**3068**(1219)
Iterations	**23.00**(5.29)	**13.00**(4.36)	**31.67**(15.53)	**6.67**(2.08)	**17.33**(2.52)	**6.67**(2.52)
Joint Effort (radians)	**202.37**(48.16)	**36.78**(3.02)	**128.09**(71.91)	**8.42**(3.66)	**75.32**(24.37)	**9.41**(4.02)
Computation Time (%)	**14.36**(2.91)	**48.72**(6.8)	**46.99**(1.68)	**91.74**(0.87)	**43.46**(4.00)	**89.65**(2.02)

TABLE II

RESULTS FROM HARDWARE EXPERIMENTS (A,B,C) AND THE SIMULATED EXPERIMENT (D). THE **BOLD** NUMBER IS THE MEAN AND THE BRACKETED NUMBER IS STANDARD DEVIATION. ALL RESULTS ARE FROM THREE TRIALS FOR EACH METHOD PER ENVIRONMENT.

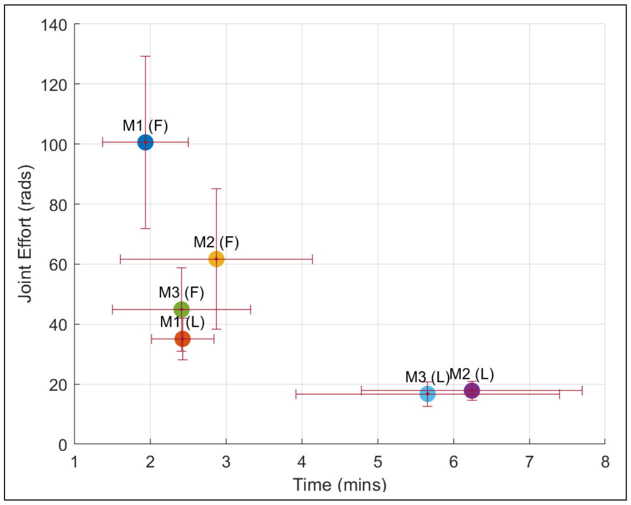


Fig. 3. Average Joint Effort (radians) vs Time (mins) for each method over all environments. The error bars represent the standard deviation.

percentage and time taken. An explanation for this relationship is that the lowest coverage utility functions (M2 (L) and M3 (L)) give priority to finding a path with the least joint effort. If the robot takes low effort paths between viewpoints, then it is less likely to observe regions far away from the frontier. The highest coverage utility function, M1 (F), has the least time taken and the largest average joint effort. This could be useful when the application selects operation time as higher priority than increased wear on the robot.

It is important to note that the average coverage percentage was different for each environment. This occurred because it is not always possible to observe the whole environment from a limited robot workspace. The portion of the volume that is not observable varies for each environment. The algorithm is attempting to maximise the coverage given a fixed robot base position.

Figure 5 shows the paths of M1 (F) and M3 (F) as a set of orange and blue lines respectively. These paths illustrate the different behaviours exhibited. The paths are superimposed upon a 3D model of the boxes environment generated by a SLAM algorithm [17] run in parallel with EGEA.

This demonstrates an ability to support other tasks, such as building a 3D model of the environment, during exploration. This figure also shows that the two methods result in differing behaviours in the robot, with the M1 (F) method producing larger distances between successive viewpoints when compared to the M3 (F) method.

VI. CONCLUSIONS

In this research, a novel approach to the frontier exploration algorithm has been proposed. This approach has been demonstrated to work in multiple environments involving real hardware and in simulation. These environments varied from small scale lab contexts to large scale industrial environments. This algorithm could be adapted for robotic platforms with different sensing and movement capabilities.

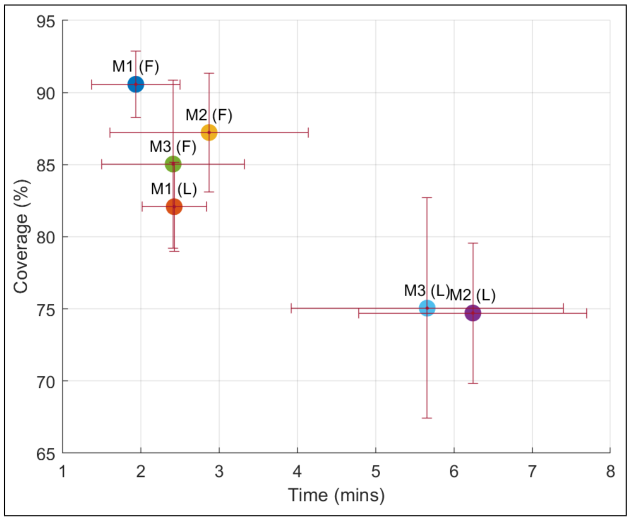
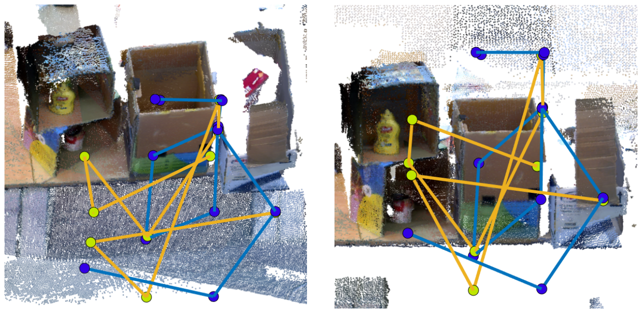


Fig. 4. Coverage (%) vs Time (mins) for each method over all environments. The error bars represent the standard deviation.



(a) Upper View (b) Front View

Fig. 5. The path of the sensor using Method 1 (orange line) and Method 3 (blue line) when exploring the box environment. The path is superimposed upon a 3D model generated using a SLAM algorithm [17].

This algorithm enabled evaluation of multiple strategies for choosing the next best viewpoint for gaining information from an environment. In particular, this algorithm has demonstrated quick exploration with a small computational load given the high dimensional nature of the problem. The algorithm has also demonstrated an ability to optimise for reduced motion of the robot with a similar level of exploration. This can result in greater operational longevity of the robot, and reduced costs and time lost due to maintenance.

Future investigations outside the scope of this work include performing a comparison study with other state of the art algorithms. Alternative methods to reduce the search space could be explored, such as choosing areas within the environment by clustering the unknown voxels. The performance of the algorithm could be measured when planning for exploration with quadcopters and mobile ground based robots. In addition, demonstrating EGEA in a real industrial environment will present opportunities to establish the strengths of this algorithm in real world scenarios.

ACKNOWLEDGMENT

The presented work was supported by the Engineering and Physical Sciences Research Council (EP/N018494/1). *This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.*

REFERENCES

[1] A. Adán, B. Quintana, and S. A. Prieto, "Autonomous mobile scanning systems for the digitization of buildings: A review," *Remote Sensing*, vol. 11, no. 3, 2019. [Online]. Available: http://www.mdpi.com/2072-4292/11/3/306
[2] A. Singh, S. H. Seo, Y. Hashish, M. Nakane, J. E. Young, and A. Bunt, "An interface for remote robotic manipulator control that reduces task load and fatigue," in *2013 IEEE RO-MAN*, Aug 2013, pp. 738–743.
[3] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, July 1997, pp. 146–151.
[4] R. Shade and P. Newman, "Choosing where to go: Complete 3d exploration with stereo," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2806–2811.
[5] P. S. Blaer and P. K. Allen, "Data acquisition and view planning for 3-d modeling tasks," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2007, pp. 417–422.
[6] J. I. Vasquez-Gomez, L. E. Sucar, and R. Murrieta-Cid, "Hierarchical ray tracing for fast volumetric next-best-view planning," in *2013 International Conference on Computer and Robot Vision*, May 2013, pp. 181–187.
[7] 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 *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, June 2015, pp. 1097–1102.
[8] P. Quin, G. Paul, A. Alempijevic, D. Liu, and G. Dissanayake, "Efficient neighbourhood-based information gain approach for exploration of complex 3d environments," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 1343–1348.
[9] J. I. Vasquez-Gomez, L. E. Sucar, and R. Murrieta-Cid, "View planning for 3d object reconstruction with a mobile manipulator robot," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 4227–4233.
[10] S. Song and S. Jo, "Online inspection path planning for autonomous 3d modeling using a micro-aerial vehicle," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 6217–6224.
[11] C. Friedrich, V. Zielke, M. Toussaint, A. Lechler, and A. Verl, "Environment modeling for maintenance automation-a next-best-view approach for combining space exploration and object recognition tasks," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, Aug 2017, pp. 1445–1450.
[12] K. Nagatani, S. Kiribayashi, Y. Okada, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, and Y. Hada, "Redesign of rescue mobile robot quince," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, Nov 2011, pp. 13–18.
[13] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," vol. 34, 04 2013.
[14] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sep. 2004, pp. 2149–2154 vol.3.
[15] I. Farquar, "AWP2018-RM-2-T004 D006 - Integrated RM CAD model," EUROfusion (IDM: 2N88G8), Tech. Rep., 2019.
[16] C. Damiani et al., "Overview of the iter remote maintenance design and of the development activities in europe," *Fusion Engineering and Design*, vol. 136, pp. 1117 – 1124, 2018, special Issue: Proceedings of the 13th International Symposium on Fusion Nuclear Technology (ISFNT-13).
[17] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.