

Towards Total Coverage in Autonomous Exploration for UGV in 2.5D Dense Clutter Environment

Evgeni Denisov¹^a, Artur Sagitov¹^b, Konstantin Yakovlev²^c,
Kuo-Lan Su³^d, Mikhail Svinin⁴^e and Evgeni Magid¹^f

¹*Department of Intelligent Robotics, Higher Institute for Information Technology and Intelligent Systems, Kazan Federal University, 35 Kremlyovskaya street, Kazan, Russian Federation*

²*Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, Moscow, Russian Federation*

³*Department of Electrical Engineering, National Yunlin University of Science and Technology, Tainan City, Taiwan*

⁴*Robot Dynamics and Control Laboratory, College of Information Science and Engineering, Ritsumeikan University, Noji Higashi 1-1-1, Kusatsu 525-8577, Japan*

Keywords: Mobile Robot, Path Planning, Autonomous Exploration and Coverage Algorithm, Next-best-view, Dense Clutter Environment, Environment Reconstruction.

Abstract: Recent developments in 3D reconstruction systems enable to capture an environment in great detail. Several studies have provided algorithms that deal with a path-planning problem of total coverage of observable space in time-efficient manner. However, not much work was done in the area of globally optimal solutions in dense clutter environments. This paper presents a novel solution for autonomous exploration of a cluttered 2.5D environment using an unmanned ground mobile vehicle, where robot locomotion is limited to a 2D plane, while obstacles have a 3D shape. Our exploration algorithm increases coverage of 3D environment mapping comparatively to other currently available algorithms. The algorithm was implemented and tested in randomly generated dense clutter environments in MATLAB.


1 INTRODUCTION


In recent years, 3D reconstruction systems have evolved towards highly-detailed and accurate reconstructions of 3D environment, which is now possible even with monocular cameras (Engel et al., 2014). Developments in parallel GPU-based computing enabled online processing of incoming sensory data, which allows moving away from offline data processing. Applications of 3D reconstruction systems include autonomous navigation in mobile robotics, 3D scanning and augmented reality applications. 3D reconstruction requires to face various challenges such as dealing with dynamic environments, small range sensing, featureless monotone and reflective surfaces,


dynamic lighting conditions etc.


Although many of the above mentioned problems still do not have robust solutions, researchers attempt to go one step ahead and develop algorithms for 3D exploration and coverage planning under assumption that existing SLAM systems are robust and close to its final development. Notable results in these scientific areas can lead to such advantages as fully autonomous 3D scanning and intelligent reasoning in 3D space. Similarly to 2D exploration path planning (González-Banos and Latombe, 2002), major issue in 3D exploration is development of a globally optimal planning algorithm (Lavrenov et al., 2017) that provides total coverage of visible environment in free configuration space without a-priori knowledge of environment (even for static environments). We further discuss those issues in Section 2.


In this paper, we present a novel algorithm that optimizes existing exploration and coverage strategies for a simple 3-DoF UGVs in 2.5D dense clutter environments. To evaluate our results, we developed environment generation algorithm that creates random


^a  <https://orcid.org/0000-0003-0868-0197>

^b  <https://orcid.org/0000-0001-8399-460X>

^c  <https://orcid.org/0000-0002-4377-321X>

^d  <https://orcid.org/0000-0003-2807-2121>

^e  <https://orcid.org/0000-0003-2459-2250>

^f  <https://orcid.org/0000-0001-7316-5664>

dense clutter environments. Additionally, we introduced a new type of voxels that we refer as "hard to observe" to formally define dense clutter environments and to provide a better tool for evaluation of algorithms in this area.

The remainder of this paper is organized as follows. Section 2 overviews related work and highlights limitations of existing solutions. Section 3 describes problem definition, system setup and our proposed solution approach. Section 4 presents simulation setup while Section 5 analyzes and evaluates simulation results. Finally, we discuss our future work plans in Section 6 and conclude the paper in Section 7.

2 RELATED WORK

The problem of 3D space exploration and coverage is fairly recent in scientific research. Heng et al. were among the first authors who introduced the idea of combining 3D space exploration and coverage problems together (Heng et al., 2015). They pointed out that majority of works (further discussed in (Heng et al., 2015)) on exploration attempted to avoid 3D space, while research works on coverage were based on a-priori available entire map of environment and next-best-view algorithms assumed only single objects of known sizes. They presented an information gain-based heuristic solution for unmanned aerial vehicles (UAVs), which relies on selection of closest frontiers with high information gain, mostly known as next-best-view gain (see Section 3.2.2). Next-best-view gain remains a highly popular tool within solutions for 3D space exploration and coverage problem.

In (Adán et al., 2015) a similar next-best-view solution was presented for 3-DoF unmanned ground vehicle (UGV) with omnidirectional sensor. Coverage planning with a known map (Dornhege et al., 2013) can be used incrementally, but requires additional planning in order to obtain time efficiency. Most widespread solution to 3D space exploration and coverage problem was proposed by (Bircher et al., 2016) with RRT* path planning algorithm (LaValle, 1998). They improved RRT* algorithm (with regard to the search results and adaptiveness to changes relatively to other existing approaches) by stopping a current best branch execution after one node. A newer version of this paper introduced an ability to explore visible space in surfaces instead of voxels (Bircher et al., 2018). (Senarathne and Wang, 2016) introduced an idea of searching edges of known surfaces instead of unobserved frontiers, which performs well for orthogonal environment. (Mendez et al., 2017) proposed to take into account paths that provide a gain in qual-

ity for modelling results inside voxels. They also made a multi-robot implementation for cooperative stereo-pair planning. (Dang et al., 2018) implemented human-inspired visual attention model that plans exploration towards visually salient areas in RGB image. (Meng et al., 2017) used genetics algorithms as an extra step to refine robot movement between selected frontier viewpoints.

Vast majority of research in this area concentrate on algorithms for UAVs because of a limited teleoperation control and low capacity batteries that demand automation for UAV-based search and coverage tasks. Even though design of these algorithms is usually platform-independent and may suit various simple robots (including 3-DoF UGVs), a few works, including this paper, design solutions specially for 3-DoF UGVs without targeting for generalized solutions.

We highlighted two limitations of the above mentioned works that in our opinion are the most important:

1. The algorithm evaluation was not performed in a highly dense clutter unstructured environment. The exception is (Zhang et al., 2017) where (unstructured) 3D structures of caves were used, but typical caves lack a density of cluttered structures and the proposed solution was not fully autonomous.

2. Low interest towards total capture of an environment as a main target of exploration. For UAVs this arises from expensiveness of viewpoint sampling in 3D free configuration space. Moreover, most papers stated that dense viewpoint sampling in 2D free configuration space, which might be necessary for total coverage, is too expensive for UGVs, and thus is ignored.

To test against these limitations for a 3-DoF UGV setup, we have created a random environment generation algorithm with a novel heuristic planner that outperforms current notable RRT*-based algorithm (Bircher et al., 2016).

The motivation behind using a dense clutter environment is that it's not clear how well existing and newly introduced algorithms would perform in complex environments. Majority of authors simply test their algorithms inside a limited set of indoor environments (including own laboratories or university buildings). When they use simulated environment it's mostly not significantly more complicated than their real working environment. But there exist a variety of significantly more complicated types of environments, which can be considered important for application of autonomous 3D exploration algorithms, especially for USAR (Urban search and rescue): caves, forests, mountains, construction sites,

partially or fully destroyed buildings, junkyards, etc. Aside from that, making dense or even simply high clutter environment for real experiments is challenging, especially if a researcher wants to add some exotic objects that come out of environment walls or hang down from a ceiling.

3 OVERVIEW

This section describes our approach in details. After formal problem definition, an overview for both path-planning and random environment generation algorithms are presented.

3.1 Formal Problem Definition

Our problem definition is similar to other works in this research field. The goal is to autonomously explore a bounded 3D space $V \subset R^3$ while minimizing total coverage time. We classify V as free, occupied, non-observed or residual voxels. Areas that are not observable from any configuration ξ by a given robot and its sensor(s) are denoted as V_{res} and the goal of exploration is considered to be achieved when $V_{free} \cup V_{occ} = V - V_{res}$.

To define dense clutter environments we specify "hard to observe" voxels or simply V_{hto} . The environment is considered densely cluttered if it contains a large amount of voxels that are observable only from a very limited number of free configuration space locations. Figure 1 demonstrates a simple example of V_{hto} concept. Green cells denote occupied voxels, red cells are unobservable voxels, white cells are observation space, grey cells are non-observation space and the yellow cells is the only voxel in the picture that is observable from a limited, but not an empty set of viewpoints.

Current state-of-the-art algorithms consider a path that can observe more unobserved voxels as the best one, which is not the case when we take total coverage as a main goal, because these leftovers from paths without highest count of unobserved voxels will lead to extra planning steps closer to the end of coverage process with huge distance costs. The fault of these approaches becomes more obvious if we consider this problem as a variation of the traveling salesman problem. In our research goal we are looking for every visible voxel (even if it could be observed only from a single free space configuration) and therefore it is important not to postpone small unobserved environment patches for a future exploration.

Aside from that, another possible application of V_{hto} concept comes from its ability to determine pro-

posed algorithms overall progress without performing total coverage exploration, because in both real and simulated experiments process may require significant (scale of hours and days) exploration time. Therefore, for evaluation of algorithms in Section V we simply count V_{hto} based on a-priori knowledge with a user-defined threshold and provide information about how many of V_{hto} were missed leftovers. Section 5 evaluates both algorithms.

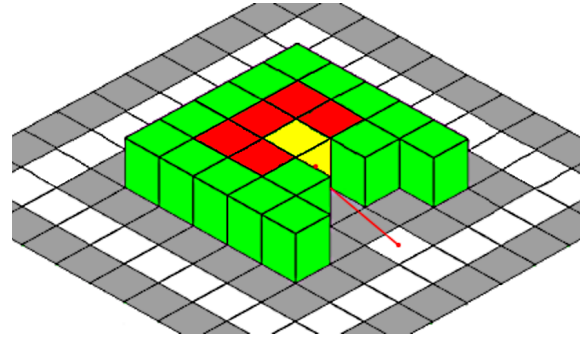


Figure 1: An example of V_{hto} concept: occupied (green) and unobservable voxels (red), observation space (white), non-observation space (grey) and the only voxel that is observable from a limited non-empty set of viewpoints (yellow).

3.2 Path-planning Algorithm Overview

To approach this problem we had to reformulate classic frontier solutions (for example, in (González-Banos and Latombe, 2002)) as they do not suited for coverage in complex 2.5D and 3D environments. First, step towards a more brute force solution than a heuristic one. Still our algorithm densely scans only viewpoints inside a limited squared area around a robot to avoid high computation costs. Every viewpoint inside this area is checked for visibility of unexplored surface voxels (V_{us}). A viewpoint with a minimum distance to the robot is considered as a next path. After reaching the best viewpoint the robot rotates towards unexplored voxels. In order not to emphasize space exploration priority over obstacle examination, we ignore floor surface voxels like if there is no obstacle between them and the robot.

After all contact space inside a scanning area is fully explored or the robot can not reach a next cell we switch to RRT* algorithm with NBV gain. A contact cell is marked as fully explored when there is not a single V_{us} that is visible through this cell in a free space within robot scanning area. Note that our algorithm searches V_{us} outside the scanning area since this area contains only viewpoints and so all V_{us} in the scanning range from every viewpoint are considered. Algorithm 1 is visualized in Fig. 2. The robot (3x3 cell size with yellow dot at the center as its depth

sensor) explores space behind contact space around (highlighted as red square area). On the left image white cells are fully explored contact space, blue cells are partially explored contact space and light grey cells are free space from which unexplored surface voxels are visible.

To improve effectiveness of our approach we added a step that produces a *limiting line* for convex regions. Without this step our algorithm tends to leave unexplored corners if searching area size is smaller than robot vision range. Limiting line is considered optimal if it is the shortest line that paths through the robot location. A cooldown feature is added to let the robot moving few extra steps before producing a next line (Fig. 3: limiting lines are shown in red, yellow line is the robot path, which is produced by our planner; other colors are the same as in Fig. 2).

```

Algorithm 1: Overview of our planner concept.
Initialization;
while termination condition not met do
    if area around robot contains not entirely
        explored contact space then
        find  $V_{hto}$  and free space;
        go to a free cell with highest gain;
        produce a limiting line if possible;
    else
        plan exit strategy with RRT*;
    end
end
    
```



Figure 2: Algorithm 1 concept (2D view from above): robot (yellow) explores space behind contact space (red square area). Left image: fully explored contact space (white cells), partially explored contact space (blue cells) and free space (light grey) from which unexplored surface voxels are visible.

Estimate of information gain for a viewpoint cell that we get after evaluation is calculated using the following equation:

$$Gain_{hto} = HTO * e^{-d} * LIMIT \quad (1)$$

, where d is the distance cost, HTO is the sign of a number of visible V_{us} from a viewpoint and $LIMIT$ is zero if a cell is placed outside of convex region, which

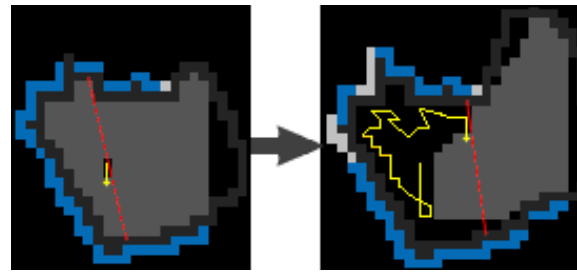


Figure 3: Example of limiting lines (red) sampling. Yellow line is the robot path.

is closed with a limiting line. Termination condition for our algorithm is a state with no empty frontiers left and all of contact space is being fully explored in 2D top-view with some error. It is possible to continue with RRT* planner if necessary.

Next subsections briefly explain the concepts of RRT* and next-best-view gain that were used in our planner's step when there is no fully examined contact space left near the robot.

3.2.1 Exit Strategies with RRT*

Rapidly-exploring random tree (RRT, (LaValle, 1998)) is a path-planning algorithm that expands a random tree towards unexplored areas until it reaches a goal. The algorithm samples a collection of connected nodes in SE(3) space without intersections of non-free space and between their connections. A branch is considered the best for execution if it has a highest next-best-view gain relatively to other nodes of the branch

$$Gain_{nbv} = NBV * e^{-\lambda d} \quad (2)$$

, where d is a distance cost, NBV is a gain from a node by the count of visible empty voxels and λ is a parameter that regulates the gain from nodes in a far distance. After the robot movement execution, the maxima of the second best branch becomes a new goal for a next random tree. RRT* algorithm improves convergence (of the original algorithm) towards a goal point. Some of the related work applied RRT* as a core algorithm. Additionally, we limit a max amount of tree nodes and stop expansion after reaching a certain number of nodes. A typical result of RRT* sampling is visualized in Fig. 4.

In our algorithm there are two exit strategies that involve RRT* path-planning algorithm with NBV gain. The first is for exploring empty environment without visible obstacles in range of a depth sensor and for exiting dead ends. The second is used in rare occasions when the robot may get stuck in a dead-end with only a narrow corridor for returning. To avoid

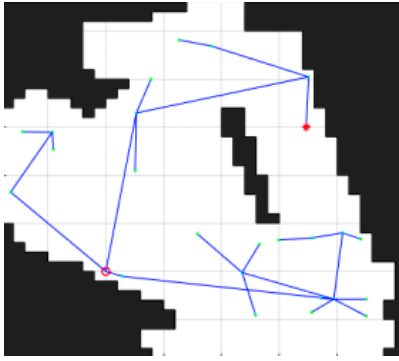


Figure 4: RRT* produces a tree with nodes as NBV view-points. The process stops when the tree reaches the end point (red dot) from the start point (red circle).

this we added another strategy with a different usage of RRT* without NBV gain: if the robot does not receive information gain from RRT* planner for a few steps then it finds a closest empty frontier and increases the limit of nodes by a large number to ensure the robot will find a way out.

3.2.2 Next-best-view Gain

Next-best-view is a class of problems that target for optimizing an environment coverage path to minimize a number of required camera shots. In addition next-best-view term is widely used as a description for views with high gain of new information about environment. Consecutively, every configuration ξ is a point of view, which represents some part of an environment by a set of visible and unmapped voxels. The most common method for gathering this data is to cast a grid of rays from a point of view with a given field of view and limited distance. For this action we use a raycast algorithm implementation for 3D voxel-based grid (Williams et al., 2005). After all rays are projected, the next-best-view gain describes a proportion between amount of occluded and not-occluded rays. NBV raycasting concept is visualized in Fig. 5.

3.3 Random 2.5D Dense Clutter Environment Generation

We propose an algorithm for generation of 2.5D cluttered environment for 3-DoF UGVs robot configuration. In this paper 2.5D restricts 3D space by prohibiting elevation in free space that is a typical structure inside buildings. Algorithm consists of two steps. Firstly, we sample onto 2D plane a random set of points and lines between them with a random width in a way that there is no single point without a connection to some other point. The result defines collision-free space. Secondly, we sample random voxels onto

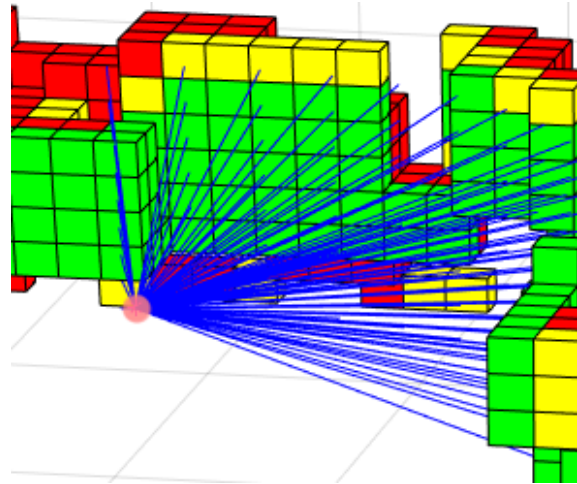


Figure 5: Next-best-view gain is defined by a set of raycasts. The density of rays is decreased for better visualisation.

3D space outside collision-free space with addition of much smaller amount of random cuboids increasing variety. For visualization we use a voxel plotting tool that is made by (Shabat and Fischer, 2015). Figure 6 presents an example of a generated environment.

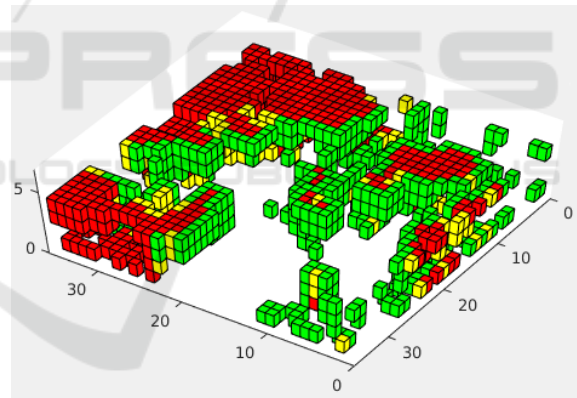


Figure 6: This figure shows a labeled generated environment. Observable voxels are green, non-observable voxels are red, "hard to observe" voxels are yellow (V_{hto}). Walls and voxels above the robot head are omitted for better visualization.

4 VIRTUAL EXPERIMENTS

For this paper, we selected a 3-DoF configuration UGV robot with position and yaw represented by $\xi = (x, y, \psi)^T$. All path-planning is done inside a collision-free space. A depth sensor is located on a fixed height from a ground plane, has a limited field of view (90 degrees in horizontal axis and 70 degrees in vertical axis) and a limited range ($R = 15$ voxels).

The depth sensor pitch is lowered to make sure the robot can see a path in front, parts of environment that are located above the robot head are not visible. It is preferable to place the camera that way in order to avoid an extensive search of a highest reachable voxel rows, but for complete scanning results we plan to utilize a high-DoF mobile robot with vertically mobile camera as a part of our future work. The robot size is $3 \times 3 \times 6$ voxels. Generated environments are represented inside a $70 \times 70 \times 6$ box. The limit for maximum number of RRT* nodes is set relatively high for 2D free space in order to increase a chance of obtaining more V_{hto} at the end of the coverage process. Convex detection range is set higher than the sensing range, but HTO scanning range is set lower to speed up computations.

The algorithm was implemented in MATLAB, except the Woo raycasting algorithm library (C++ code), which was compiled as a mex64 file in order to speed up the computations. We assume an ideal 3D reconstruction in our setup to achieve persistent estimations for equivalent experiment settings. The algorithm performance was evaluated with regard to a RRT* planner (Bircher et al., 2016), although we dismissed the idea of cutting off a current best branch on the first node as it leads to worse performance in our dense clutter environment than without it. Other frontier algorithms were considered, but they lack the ability to determine unreachable frontiers or can ignore actually reachable frontiers while randomly sampling viewpoints.

We took 6 generated maps (Fig. 7) from our random environment generator. The maps were chosen on the variety of quantity of V_{hto} they have and topological differences. In every map the robot starts at the left bottom corner. Due to stochastic nature of the RRT* algorithm, we additionally included 3 extra trials for both planners on every map. Termination condition for both planners in these simulation experiments was set as a coverage of 97 % and usually HTO planner explored all walls up to this point, so after 97 % both algorithms worked in the same way. The algorithms parameters are presented in Table 1.

5 RESULTS

We evaluated performance of both algorithms comparing odometrical distance, computation time and coverage (both V_{hto} and V_{occ}). Rotation was also included in the distance metric. For evaluation we assumed voxel's size of 0.1m. Possible coverage for a map was measured with the next-best-view method with 360 degree angle from free space grid with dou-

Table 1: Parameters for algorithms. 'v' stands for voxels.

Parameter	Value
Sensor range	15 v
Robot's height	6 v
Robot's size	3×3 v
RRT* tree edge	6 v
λ for RRT*	0.4
RRT* max nodes	25
HTO scanning area	20×20 v
Convex detection area	40×40 v

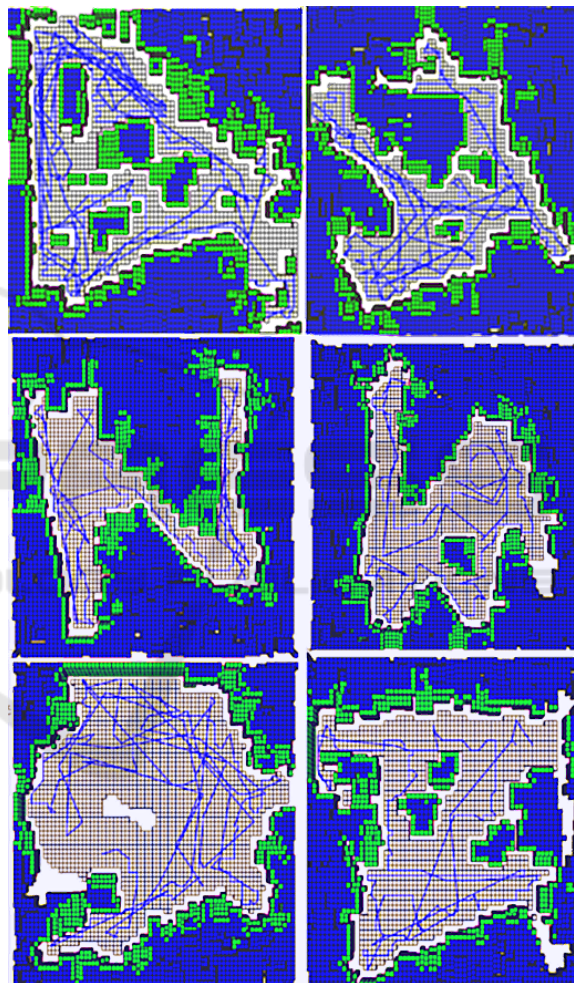


Figure 7: All 6 generated maps after full or partial exploration. Blue voxels are both unobserved and unobservable voxels from ground truth and green voxels are explored voxels. Blue lines are taken paths in exploration.

bled density. V_{hto} threshold was set to 9 which leads in selected maps to 5-10 % of total coverage.

We run total coverage tests on several maps and it took our planner to reach 96-97 % coverage with fully explored obstacles (stated by our planner) in 600-900 s computation time, while for RRT* planner it took

1100-1600 s in average to obtain the same coverage level (except for the most simple map that was once completed in 601 s) The measurements did not take into account huge distance penalties, which resulted in extra thousands of meters (above 2000 m), while our algorithm rarely went above 1000 m. An example of the resulting paths by both planners are presented in Fig. 8.

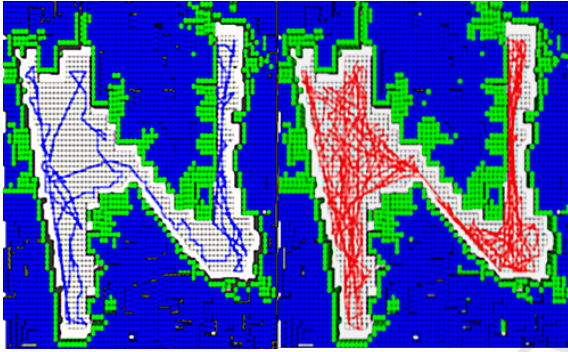


Figure 8: An example of the resulting paths of our planner (blue lines, on the left) and RRT* planner (red lines, on the right). Both planners started at the left-bottom corner of the map.

Total coverage measure of 100 % was not achieved because we only check visibility of surface voxel's central point, while next-best-view method uses raycasting visibility check for voxel's edges. Remaining 3 % in most cases also contains more than 8 % of V_{hto} . After checking total coverage we tried employing our V_{hto} metric as a prediction tool. Figure 9 shows that in most cases it is useful to define an overall success of a selected algorithm rather than an overall coverage. The plot on the right demonstrates that our planner outperforms RRT* planner by V_{occ} and V_{hto} coverage rate and we link this success to total coverage success.

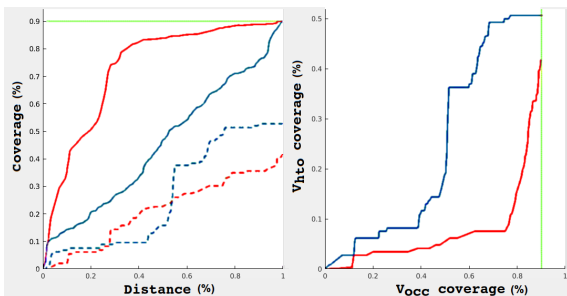


Figure 9: V_{hto} coverage (dashed line) of our planner (red) and RRT* planner (blue). The figure shows the worst case scenario for our planner. RRT* planner's V_{hto} coverage is generally worse and its coverage process is uneven.

6 FUTURE WORK

Our next goal is to further improve the proposed algorithm. Firstly, currently we decided to accept a huge rotation penalty it is much less than distance penalty. Secondly, the shortcoming of limiting convex regions is that this step ignores non-convex regions, which are also important. Thirdly, although our planner outperforms RRT* planner in minimizing traveled distance, it is more computationally demanding and requires extensions for robots with movable sensor in 3D space (UAVs or High-DoF UGVs). Fourthly, there is a lack of reliable parameters in order to deal with random obstacles. We attempted to find more parameters to improve quality of the resulting path-planning in addition to restricting the robot within convex regions, but the obtained solutions did not seem to work better in general cases for dense clutter environment. Therefore, an omnidirectional scanner would better suite for our solution, but we do not limit the application of our algorithm solely to this solution. Fifthly, the random environment generator needs to be improved towards a real 3D environment or (as a starting point) should include obstacles above the robot's head, which would require major changes in our algorithm.

Another goal, which is more technical than the above mentioned improvements, is to port the both planning and random environment generation algorithms to C++ and to integrate them into Robotic Operating System framework and to enable its functionality in Gazebo simulator (Afanasyev et al., 2015). The random environment generator will be modified to export generated models as 3D models for further simulation experiments in Gazebo, which provides actual physical simulation. Then we will re-evaluate current restrictions that were taken for MATLAB simulation to be more suitable for real experiments. In addition, we are interested to implement an instrument of automatic map generation (similar to a convenient automatic map generation tool (Lavrenov and Zakiev, 2017)) of dense clutter environments for further autonomous exploration by a UGV in order to provide a convenient benchmark for various coverage algorithms comparison.

We believe that V_{hto} metric could be a useful tool for real world experiments, where it is possible to check for hard to observe locations of an environment and then check their representation in reconstruction manually by scanning only those parts instead of taking an entire scan of the environment. After all, we consider expanding the scope of our current work to face more difficult problems, e.g., such as dynamic environment and multi-robot exploration.

7 CONCLUSIONS

In this paper, we presented a novel algorithm for autonomous exploration and coverage problem for UGVs in 3D dense clutter environment. The algorithm plans the exploration path towards finding specific voxels that are hard to observe. Our approach provided a solution that is more robust for 2.5D dense clutter environment with highly non-linear contact-space comparatively to currently known solutions, as in practice they largely ignore such features as contact space and narrow passages, while our algorithm is opportunisticly looking for such locations.

The MATLAB code is available for academic community for research and educational purposes in our Gitlab repository¹.

ACKNOWLEDGEMENTS

The reported study was funded by the Russian Foundation for Basic Research (RFBR) according to the research project No. 19-58-70002. Part of the work was performed according to the Russian Government Program of Competitive Growth of Kazan Federal University.

REFERENCES

- Adán, A., Quintana, B., Vázquez, A. S., Olivares, A., Parra, E., and Prieto, S. (2015). Towards the automatic scanning of indoors with robots. In *Sensors (Basel)*. 2015 May; 15(5): 11551–11574.
- Afanasyev, I., Sagitov, A., and Magid, E. (2015). ROS-based SLAM for a Gazebo-simulated mobile robot in image-based 3D model of indoor environment. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 273–283. Springer.
- Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2016). Receding horizon "next-best-view" planner for 3d exploration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.
- Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2018). Receding horizon path planning for 3d exploration and surface inspection. In *Autonomous Robots, February 2018, Volume 42, Issue 2*, pp 291–306.
- Dang, T., Parachristos, C., and Alexis, K. (2018). Visual saliency-aware receding horizon autonomous exploration with application to aerial robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- Dornhege, C., Kleiner, A., and Kolling, A. (2013). Coverage search in 3d. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.
- Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In *IEEE European Conference on Computer Vision (ECCV)*, 2014, pp 834–849.
- González-Banos, H. H. and Latombe, J.-C. (2002). Navigation strategies for exploring indoor environments. In *The International Journal of Robotics Research*, vol. 21, no. 10–11, pp. 829–848, 2002.
- Heng, L., Gotovos, A., Krause, A., and Pollefeys, M. (2015). Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
- LaValle, S. M. (1998). Rapidly-exploring random trees a new tool for path planning. Technical report.
- Lavrenov, R., Matsuno, F., and Magid, E. (2017). Modified spline-based navigation: guaranteed safety for obstacle avoidance. In *International Conference on Interactive Collaborative Robotics*, pages 123–133. Springer.
- Lavrenov, R. and Zakiev, A. (2017). Tool for 3d gazebo map construction from arbitrary images and laser scans. In *2017 10th International Conference on Developments in eSystems Engineering (DeSE)*, pages 256–261. IEEE.
- Mendez, O., Hadfield, S., Pugeault, N., and Bowden, R. (2017). Taking the scenic route to 3d: Optimising reconstruction from moving cameras. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.
- Meng, Z., Qin, H., Chen, Z., Chen, X., Sun, H., Lin, F., and Ang Jr., M. H. (2017). A 2-stage optimized next view planning framework for 3-d unknown environment exploration and structural reconstruction. In *IEEE Robotics and Automation Letters, Volume 2, Issue 3, July 2017*.
- Senarathne, P. G. C. N. and Wang, D. (2016). Towards autonomous 3d exploration using surface frontiers. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2016.
- Shabat, Y. B. and Fischer, A. (2015). Design of adaptive porous micro-structures using curvature analysis for additive manufacturing. In *the 25th CIRP Design conference. 2015, Haifa, Israel*.
- Williams, A., Barrus, S., Morley, R. K., and Shirley, P. (2005). An efficient and robust ray-box intersection algorithm. In *SIGGRAPH '05 ACM SIGGRAPH 2005 Courses, Article No. 9*.
- Zhang, G., Shang, B., Chen, Y., and Moyes, H. (2017). Smartcavedrone: 3d cave mapping using uavs as robotic co-archaeologists. In *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017.

¹https://gitlab.com/LIRS_Projects/Simulation-3d-reconstruction/tree/master/autonomous_exploration_and_coverage