Complete Coverage Planning in Nuclear Environments Using Off-Center Proximity Sensors with 3D Obstacle Geometry

Alex Navarro, Mitch Pryor

I. Introduction

Efficient and effective monitoring of nuclear environments is critical to maintaining the safety, productivity, and security of nuclear worksites. The way in which routine radiation surveys are conducted is dependent upon the type of contamination which is being scrutinized. Gamma radiation, for instance, can be monitored through the use of personal and wall mounted dosimeters to keep track of dose rates for radiation workers. Alpha radiation, on the other hand, provides a more significant challenge to monitor, as it is undetectable at distances greater than a few centimeters. For this reason, routine surveys for alpha radiation are normally carried out by wiping surfaces with cloth and then applying the contaminated cloth to a close range alpha detector. This slow procedure causes manual alpha radiation survey to be time-consuming, labour intensive, and difficult to keep track of over long periods of time.

An alternative method for alpha radiation survey is to pass an alpha radiation detector close to surfaces of interest, reading data directly from the source. This requires the detector to stay within a small offset of the target surface for extended periods of time to develop the desired count uncertainty. This precise, long running method is ill-suited to human applications, but plays to the strengths of mobile robots. By suspending an alpha radiation detector a fixed distance from the target surface, a mobile robot can record detailed radiation count data which can be chronicled for long term analysis.

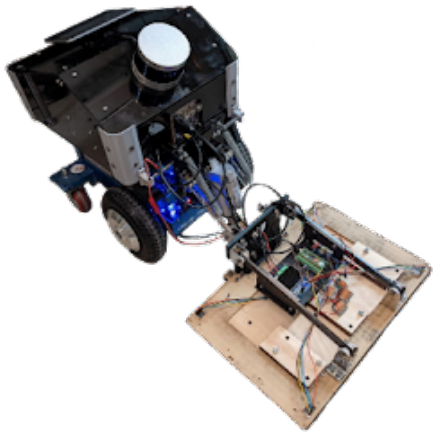


Fig. 1: Target deplayment platform is a modified Ubiquity Magni Robot with Modified Payload Deck [10]

Such an application requires the robot to have a software package which can create a motion plan that simultaneously maximizes the coverage of the survey, and minimizes the total completion time. Such an application falls under the umbrella of Complete Coverage Path Planning (CCPP). The goal of CCPP is to create a plan for the robot which visits as much of the task space as possible in an efficient manner. CCPP can be divided into online planning and offline planning [5]. Online planning generally assumes that the robot does not have access to complete environmental data, and therefore plans each step of the coverage plan as required at runtime. Conversely, offline planners assume complete knowledge of the environment, and compute the entire plan prior to execution.

The difficulty present in the problem of alpha radiation survey is that the coverage tool (the sensor) is not concentric with the robot base. Many offline and online algorithms present in the CCPP literature generate paths for the robot center, planning a complete path for the *robot* to visit the entire workspace, and not an offset sensor. Creating a complete coverage path for an offset sensor provides additional challenges such as more restrictive non-holonomic movement constraints and a long, asymmetric robot profile. However, it also provides a unique opportunity for optimization by allowing the low-profile sensor to be inserted into areas where the robot body itself would not fit. The main contribution of this work, therefore, is to provide a navigation framework which allows CCPP to be performed for an offset tool, while still allowing traditional online and offline CCPP algorithms to be applied to the coverage problem.

II. Prior Work

Coverage planning is not a new topic in robot navigation. Galceran and Carreras [5] performed a comprehensive survey in 2013 comparing a multitude of different coverage algorithms for both 2D and 3D task spaces. Their survey examined over 35 papers covering various algorithms used in coverage planning. Methods include cell decomposition [3], [1], grid decomposition [13], as well as landmark planning [11] and neural network planning [12].

For both online and offline planning problems, the task space is typically subdivided into regions which are considered obstacles or free space. The cell decomposition method breaks up the space into irregularly shaped cells which can easily be covered with primitive motions such as a back-and-forth or zig-zag pattern. These regions may be formed by polygonal obstacle approximations [3] or they may be generated on arbitrary obstacle shapes [1]. Another method uses grid-based decomposition, which is popular in general

motion planning [9], to divide the workspace. Planning can then be performed by performing spiral motions in each grid cell [4] or through propagating a wavefront through the grids and following the path of shallowest descent [13].

Cellular decomposition methods have the advantage of requiring comparatively little computer memory and generating intuitive coverage plans. However, these methods require complete knowledge of the environment and perform best in areas that are mostly comprised of free space, and so are best applied to offline planning problems in open areas. On the other hand, grid based methods are memory intensive and can result in erratic plans, but allow for flexibility in cluttered environments through the grid resolution and shape.

Grids used for coverage planning do not need to be rectilinear, such as the case of the online coverage algorithm presented by Kan, Teng, and Kardis [7]. This method uses a hexagonal grid which can be covered by circular motions of a non-holonomic robot, and plans for smooth transitions between adjacent cells. Like many grid based algorithms, prior knowledge of the environment is not required, as the grid can be populated in real time. Due to these advantages in flexibility, a grid based approach will be used in the development of the nuclear survey coverage planner.

Bormann et al [2] addressed the specific case of an off-center coverage tool, providing a solution for a floor cleaning vacuum robot. Their solution detailed first solving the coverage plan for the sensor as if it was the robot center, and then subsequently calculating suitable robot poses around each of these way-points to generate the coverage path [2]. This approach works well in a 2D case, however this does not utilize the sensor's smaller size in 3D environments. Additionally, it can create an impossible path if the sensor passes through a narrow passageway which the robot cannot follow, since there are no continuous paths which allow the robot to follow the sensor path in that case.

III. Implementation

The robot is assumed to have have access to 3D environmental knowledge at runtime, either through sensor data or a stored 3D map. For our implementation, the environment is represented using a dense pointcloud. We further assume that dynamic obstacles are only present in an online context and that all information of the environment is up to date for an offline context. Online obstacle data is stored in memory as a local 3D map through the use of custom ray-tracing software.

One of the main issues in planning for an off-center detector is that the effective navigation space for the detector is distinct from the navigation space of the robot body. That is to say, the sensor may be able to safely occupy a location which would result in a collision for the robot body, and vice-versa.

To alleviate this issue, collision detection for the robot is performed in multiple steps to allow the alpha detector, which has a much smaller profile than the rest of the robot, to be considered separately from the rest of the robot. The robot assembly is subdivided into cylindrical sub-bodies, which each correspond to a physical portion of the robot. An example subdivision is shown in Figure 2, where a total of 4 subdivisions have been used. Cylinders are used as they are rotation invariant, and keep collision calculations tractable for different robot orientations.

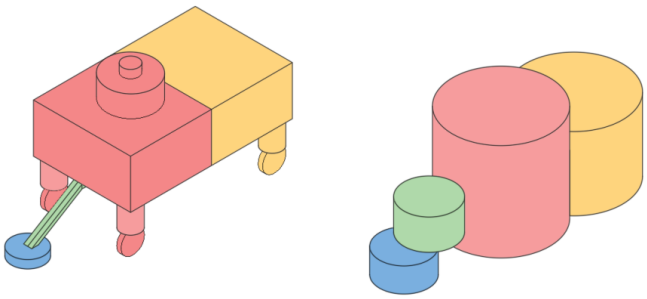


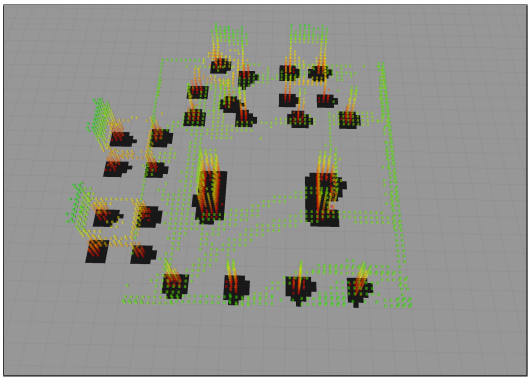
Fig. 2: Cylinder Decomposition of an Example Robot

For each of these subdivisions, a separate obstacle avoidance costmap is created. Each costmap only accounts for obstacles which would result in a collision for the corresponding robot portion, as in Figure 3. This takes advantage of the fact the wheeled mobile robots are constrained by gravity in the vertical direction, and so planar slices of the environment are all that is required to perform 3D collision detection. Memory usage is identical to that of a typical grid navigation problem multiplied by the number of cylinders. Collision checking involves performing a random-access grid lookup for each of the objects in O(1) time complexity relative to the size of the environment and number of obstacles.

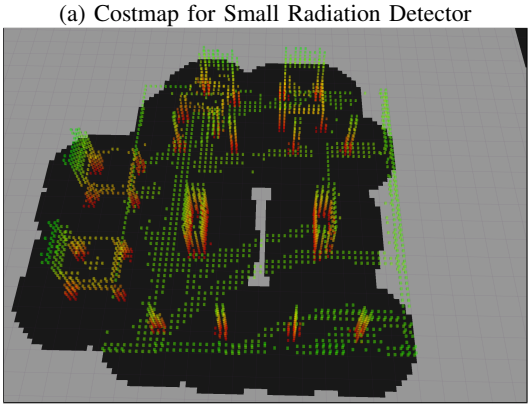
In order to create a coverage plan using these multiple occupancy grids, all of the cylindrical objects need to be considered in parallel. In order to achieve this parallel obstacle avoidance, we introduce the idea of "cell partitions" to transform this navigation problem into one to which traditional methods can be applied. A cell partition is an angle measurement representing a continuous region around a grid cell in which the robot can exist without collision. If the radiation detector was placed at the center of the coverage cell and at an angle within the range of the partition, the robot is guaranteed to be collision free, as seen in Figure 4.

These cell partitions can be connected in a graph-like structure, illustrated in Figure 5, which allows traditional navigation algorithms to be applied to this multi-costmap collision model by considering graph-adjacent partitions to be analogous to cells in a grid. To determine if two partitions are connected, an interpolated motion from one pose to the next is simulated to account for non-holonomic constraints, and collisions are checked on all costmaps at each simulation step. If no collisions are detected then the partitions are connected.

Performing these frequent simulations can put strain on the computational power of the robot computer, despite the fact the each individual transition test is computationally cheap. However, each partition generation and connection step between partitions is independent of all others, and so



(a) Costmap for Small Radiation Detector



(b) Costmap for Base Link of the Robot

Fig. 3: Two costmaps generated for different robot subdivision parts. Aside from elevation, different objects have different inflation radii to account for their sizes

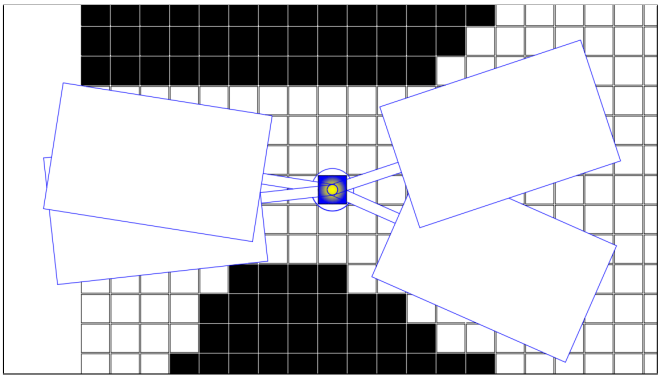


Fig. 4: Robot poses at the extremes of the two valid partitions of this coverage cell. For all orientations between each of the pairs of poses, the robot configuration is collision free

presents a perfect candidate for multi-threading. Utilizing the powerful GPU capabilities of the onboard Jetson, this process has been implemented in CUDA C++ and presents very little computational overhead at runtime. With this computational boost, the algorithm described can be run either online or offline depending on the knowledge of the environment. In an offline context, the entire grid is partitioned at once, and then all connections between partitions are made prior to execution. In the online context, only the cells in a local area surrounding the robot are partitioned and connected, and only this area is only extended if no immediate unexplored cell is found.

The coverage algorithm itself works as a variation of the Path Transform method of [13], in which a wavefront is propagated through the cells of the grid starting with a value of planned end location, and increasing by one at each iteration of the wave (Fig. 5). By then following the path of shallowest descent, the robot passes through all grid cells such that it surveys the end location last. For the proposed algorithm, the grid cells are replaced by cell partitions. Since they form a connected graph structure, the completeness condition of the original algorithm holds. Additionally, since partitions represent partially allowable locations for the robot, this algorithm also ensures that the radiation sensor will cover all reachable areas, including those requiring insertion movements to reach. For online planning, only the first step of the local coverage plan is taken, and then a new plan is recalculated at the next step.

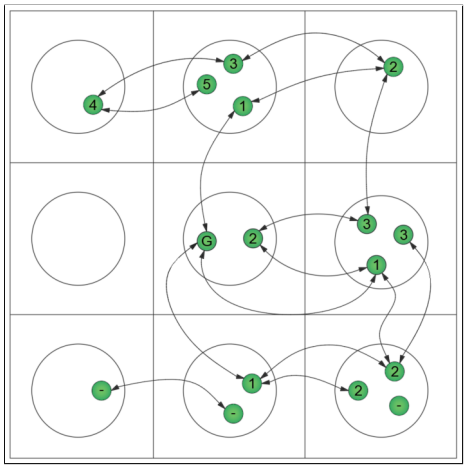
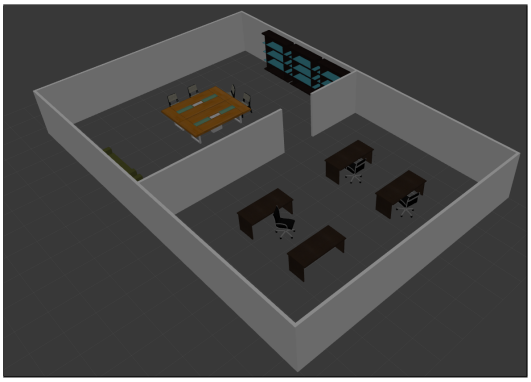


Fig. 5: Example wavefront propagation using a distance transform

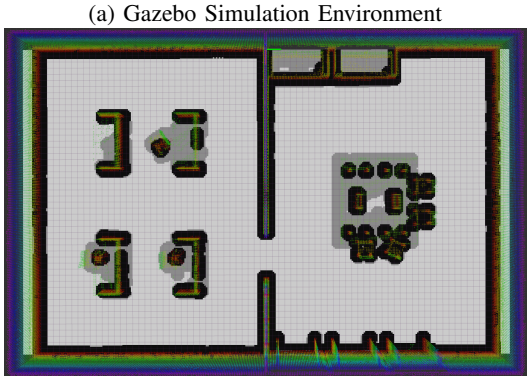
IV. Experiments and Results

The offline version of the algorithm was run in simulation using the Gazebo robot dynamic simulation environment [8]. The environment was modeled to have challenging geometry such as overhangs from desks, and closely packed obstacles such as the legs of the conference table. A 3D map of the environment was created using the Octomap software package [6], which was used as the input to the coverage planner. Figure 6b shows the obstacle representation of the workspace. The planner distinguishes between areas which are unreachable by the radiation detector, and areas which are unreachable only to the robot body.

Local planning was not performed in this simulation, instead the simulated robot was made to follow the coverage plan exactly. This means that while the robot perfectly executes the coverage plan, total execution time and computational load data are not available since the robot did not spend time planning for local obstacle avoidance.



(a) Gazebo Simulation Environment



(b) Multiple occupancy grids of the simulation environment for different parts of the robot. Black areas represent collisions for the alpha detector, while gray areas represent collisions for the robot body

Fig. 6: Simulation Environment and its Representation in the Planner Context

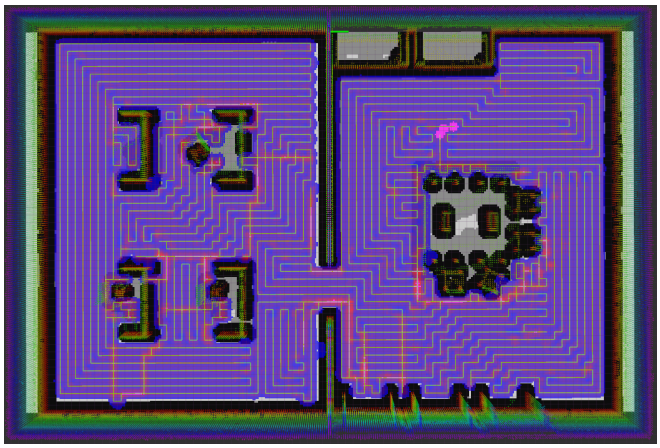


Fig. 7: Executed Coverage Plan in Simulation. Color shading represents the total time spent over each grid cell. Blue cells were surveyed for shorter periods, while pink cells had the maximum duration

For a non-holonomic test robot, the coverage plan was completed in 3011 steps in a grid of 4000 cells. Running the algorithm with the CUDA extension decreased the coverage planner runtime by 92%, improving from 5.02 seconds to only 0.39 seconds. Roughly 76% of computation time was performed on the GPU, representing the significant increase in performance over a CPU-only version. 57% of the total computation time was used in the simulation step to determine valid transitions between partitions. Figure 7 shows the results of the simulated coverage plan, indicating how well the area was covered. Compared to the obstacle representation in Figure 6b, the sensor does indeed cover areas which represent collisions for the robot body. Additionally, the red traces in Figure 7 indicate the areas where the robot had to backtrack, highlighting that very little time was spent covering areas which had already been surveyed.

Testing with the online version of the planner was not completed due to incompatibilities between the global plan created by the dynamic coverage planner, and the off-the-shelf local planner options available as part of the ROS navigation stack. This is due to inconsistent perceptions of which poses result in a collision to the robot between the global and local planner. The traditional local planners only account for the projected costmap of the 3D LiDAR scan, which means that certain poses which are commanded by the coverage planner were not feasible to the local planner. This will be remedied in a future work through the development of a Dynamic Window Approach (DWA) local planner using the obstacle representation presented in this paper.

V. Conclusion and Future Work

The concept presented is a flexible navigation framework for use in nuclear environments. In order to maximize coverage, it was determined that the robot body and the radiation detector must be planned for separately, but in parallel. This allows the detector to provide maximum coverage while maintaining a collision free path for the robot. A test case in a simulated environment was performed, and efficient coverage was realized and executed. The proposed framework is agnostic to robot shape and size, and is valid for both holonomic and non-holonomic robots through the simulation step to connect partitions.

Through the use of GPU parallelization, the computation time was reduced by over 90% compared to a CPU-only implementation. With this parallel framework, a real-time application of the framework was developed to run alongside an online coverage planning algorithm. This online addition allows for the global map to contain stale data and dynamic obstacles, without a significant effect on total coverage in theory. With the need to recreate global maps removed, the robot could then perform these radiation surveys completely autonomously, contributing much in the way of reduced manpower and increased safety in nuclear worksites.

True implementation of this framework presents multiple challenges related to non-simulated hardware. Real environments are dynamic, with small objects such as chairs and tools moved between surveys. However, due to local planner limitations, we were unable to test the framework in a such a dynamic environment. Additionally, the planner creates coverage paths which pass very close to obstacles, so a lot of effort would need to go into tuning a motion planner that performs well in extremely close proximity to

obstacles. Future experiments on the Magni robot (Fig. 1) will attempt to tackle these issues and others that will arise in the transition from simulation to actual execution.

REFERENCES

[1] Ercan U. Acar, Howie Choset, Alfred A. Rizzi, Prasad N. Atkar, and Douglas Hull. Morse decompositions for coverage tasks. *The International Journal of Robotics Research*, 21(4):331–344, 2002.

[2] Richard Bormann, Florian Jordan, Joshua Hampp, and Martin Hägele. Indoor coverage path planning: Survey, implementation, analysis. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1718–1725, 2018.

[3] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. In Alexander Zelinsky, editor, *Field and Service Robotics*, pages 203–209, London, 1998. Springer London.

[4] Y. Gabriely and E. Rimon. Spiral-stc: an on-line coverage algorithm of grid environments by a mobile robot. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 954–960 vol.1, 2002.

[5] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.

[6] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees, 2013.

[7] Xinyue Kan, Hanzhe Teng, and Konstantinos Karydis. Online exploration and coverage planning in unknown obstacle-cluttered environments. *IEEE Robotics and Automation Letters*, 5(4):5969–5976, 2020.

[8] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, 09 2004.

[9] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121, 1985.

[10] Ubiquity Robotics. Magni robot. `https://www.ubiquityrobotics.com/products-magni/`, 2022.

[11] S. C. Wong and B. A. MacDonald. A topological coverage algorithm for mobile robots. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 2, pages 1685–1690 vol.2, 2003.

[12] S. X. Yang and C. Luo. A neural network approach to complete coverage path planning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):718–724, 2004.

[13] A. Zelinsky, R.A. Jarvis, J. C. Byrne, and S. Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *In Proceedings of International Conference on Advanced Robotics*, pages 533–538, 1993.