Structural Inspection Planning for a Mobile Manipulator Robot

Andrew Washburn, Prateek Arora, Nikhil Khedekar

December 2020

Abstract

This report details a robotic coverage planning algorithm that completely inspects a 3D object using an autonomous mobile robot with an attached arm and a RGBD camera on the end-effector. Given a model of the structure to be inspected, and a respresentation of the environment, a sampling-based coverage algorithm provides a near optimal number of feasible viewpoint poses for the robot's end-effector that see the entire structure. Finding a minimal set of base positions for the mobile robot is derived from hierarchial clustering of the viewpoint positions such that one base position can reach multiple viewpoint positions. This algorithm is packaged as a ROS service and evaluated in simulation by inspecting a complicated object.

1 Introduction

Structural inspections using autonomous robots require a type of path planning called *coverage planning*, which is a subset of the more general field of *path planning* [4]. A structure can be completely seen if there are enough, or many, positions around the structure such that viewing from all of these positions will see the entire structure. Thus, the problem solved by coverage planners is to find an optimal set of viewing positions that completely see a structure or workspace and then navigating to all of the viewpoints. Computing a set of optimal viewpoints can done by decomposing the structure or workspace into uniform cells and then finding a path that sweeps through all the cells [1]. An initial solution can be further optimized for shortest path length [3] or number of unique viewpoints [2].

Much of the research in structural inspection use unmanned aerial vehicles (UAV) [3] or underwater robots [6]. These convenient platforms can move with 6 Degrees of Freedom (DoF) wherever the air or water allows them to, which puts few constraints on reachable viewpoints, or feasible paths. However, our work focuses on a skid-steer mobile robot with an mounted robotic arm and a RGBD camera attached to the end-effector. For inspections, this platform is advantageous for observing objects with hard to view surfaces such as an elevated tank see in Figure 1. The robotic arm can extend underneath the tank and capture images, whereas a quadrotor UAV would not be able to reach underneath because of the unsafe nature owing to the possibility of hitting the tank with its blades. Because our mobile robot is more restricted than UAVs, we cannot use the same coverage planning algorithms. So we propose a new complete coverage planner given a mobile robot's constraints.

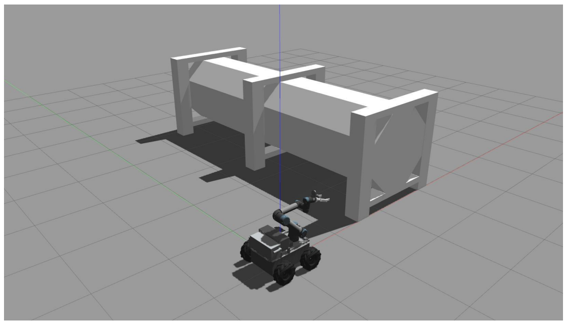

Figure 1: The mobile manipulator and structural object to inspect.

2 Coverage Algorithm

Our coverage planning algorithm derives from the sampling based coverage planner proposed and implemented by Gonzáles-Baños [2] and Danner [5]. An ideal coverage planner is *probabilistically complete*, meaning that if an entire structure could possibly be viewed without any limiting constraints, then as the number of viewing positions increases the more likely the entire structure has been seen [6]. The planner assumes coverage is only performed on the nodes of a graph, not the edges. The coverage problem can be stated as a *set system.*

Definition 2.1 (Set System) *Given a polyhedron model of an object P made up of points p_i, find within the robot's configuration space $\mathcal{Q}$ a set of viewpoints $q_j \in \mathcal{Q}$ such that all $p_i \in P$ are observed.*

Every configuration q_j views a subset of p_i. Given the possible viewpoints and the subsets of p_i that are viewed by them, the *set cover problem* finds the minimum number of configurations in $\mathcal{Q}$ such that every $p_i \in P$ is viewed. Figure 2 shows a viewpoint q_j observing multiple points $p_i \in P$ on the structure.

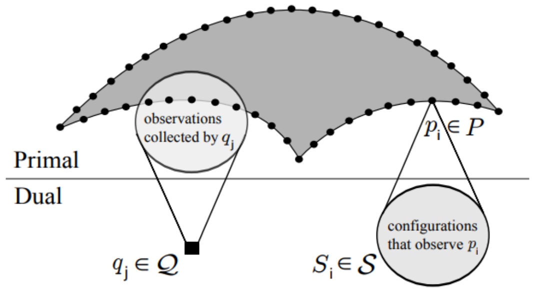
observations collected by q_j
$p_i \in P$
Primal
Dual
configurations that observe p_i
$q_j \in \mathcal{Q}$
$S_i \in \mathcal{S}$

Figure 2: The coverage sampling problem primal and dual set system [6].

Another view of the set cover problem is the *dual set system.*

Definition 2.2 (Dual Set System) *Every point p_i can be seen from a set of viewpoint configurations $S_i \in \mathcal{S}$ that exist in the robot's configuration space $\mathcal{Q}$.*

Figure 2 shows the configuration space S_i that can view p_i. The viewpoints within S_i observe other points in the structure around p_i, so different subsets are not mutually exclusive. The goal is to find a minimum set of configurations $\mathcal{G}$ such that at least one configuration lies in each viewpoint set S_i for all $p_i \in P$. Figure 3 shows the shape of set space S_i constrained by incidence angle τ, min and max distance. Another constraint is whether the robot's end-effector can reach the position.

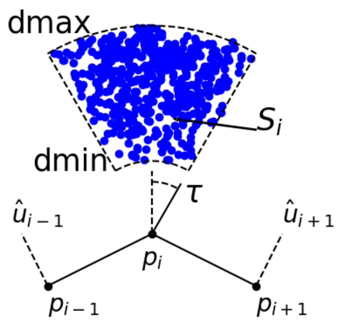

Figure 3: Random sampling the configuration space S_i. This figure shows how the distance and angle constraint form a clipped cone.

2.1 Random Sampling Viewpoints

Each region S_i is computed with a random sampling process from each point in the object $\mathcal{O}(p \in P, constraints)$ described by Latombe [2]. First we sample a point s within a rectangular region of height $dmax$ and width and depth from $\pm dmax \sin(\tau)$. The first constraint checks if s is between $dmin$ and $dmax$ and lies within the incidence angle τ as shown in Figure 3. The formula for calculating an angle θ uses the dot product between s and another vector u:

$$\theta = \cos^{-1}\left(\frac{s \cdot u}{|s||u|}\right)$$

For checking the incidence angle u is an upward facing unit vector along the z-axis.

Next, transform s into the reference frame of p_i and its unit normal direction $\hat{u}_i$. The normal direction defines the outward face of the point on the polygon. First s makes a 3D rotation about an axis of rotation calculated by $\hat{z} \times \hat{u}_i$ and by the angle between $\hat{z}$ and $\hat{u}_i$.

Now that s is in the position relative to the structure, we check if the robot can reach the point. The configuration space $\mathcal{Q}$ is limited by where the robot can freely move $\mathcal{W}_f$ and reachable by the arm length ℓ. We model $\mathcal{W}_f$ with complete a probabilistic road map [7, 9] that samples the free floor space around the object P and only keeps samples that are some distance away from the object. Our sample criteria keeps points that are at least the robot radius r distance away. Given the free workspace $\mathcal{W}_f$ as a KD-Tree, and the sample point s, if the distance between s and the closest point in $\mathcal{W}_f$ is less than the arm length ℓ then s is considered feasible.

At this point s is added to the set S_i. the algorithm generates $\mu = 500$ samples for each configuration space S_i. The next step finds an optimal viewpoint within this set.

2.2 Viewpoint Greedy Optimization

The optimal viewpoint $g \leftarrow OPT(S_i, \mathcal{X})$ collects the most new observations in the object P. The list $\mathcal{X}$, initialized $\forall\ p_i \in P$, stores the unseen points. Each iteration, all new visible observations $p \leftarrow \mathcal{V}(g; \tau, dmin, dmax, fov)$ are removed from $\mathcal{X}$. The coverage algorithm terminates once $\mathcal{X}$ is empty.

The best viewpoint optimization iterates through all possible viewpoints $s \in S_i$ and computes the number of visible points from that pose. Visible boundary points must be within the field of view fov angle from q, within a maximum distance $dmax$ and the boundary must face towards s within an incidence angle τ, just like when computing S_i. The viewpoint unit direction is $\hat{d} \leftarrow p_i - s$.

We also check if any boundary facet is between s and p using a rasterization algorithm [10]. Algorithm 1 details the steps. First, the distance t is calculated from the viewpoint s along its direction $\hat{d}$ to each triangluar facet $f_i \in \mathcal{F}$. The intersection point on the facet from the viewpoint is $p_i \leftarrow s + d * t_i$. Since p_i is in the same plane as f_i, an edge function determines if p_i is on the left or right side of the triangle edges. Figure 4 shows p_i if inside the facet if the cross product between the facet edges and the projected point $e_i \times p_i$ is all positive. If any facet closer to the viewpoint than the point we're trying to see, then the viewpoint cannot see the point and Algorithm 1 returns False.

The rasterization algorithm complexity is very expensive. Every point p_i , $O(n)$, has a sample space with μ, $O(\mu)$, viewpoints checking if each visible point p, $O(n)$, has any facet in front of it, $O(n)$. The asymptotic function is $O(\mu n^3)$, where μ is the number of samples in S_i and n is the number of facets in the object P. We noticed the algorithm running time increases from 30 seconds to 10 minutes on a laptop with an Intel i7 2.4GHz processor!

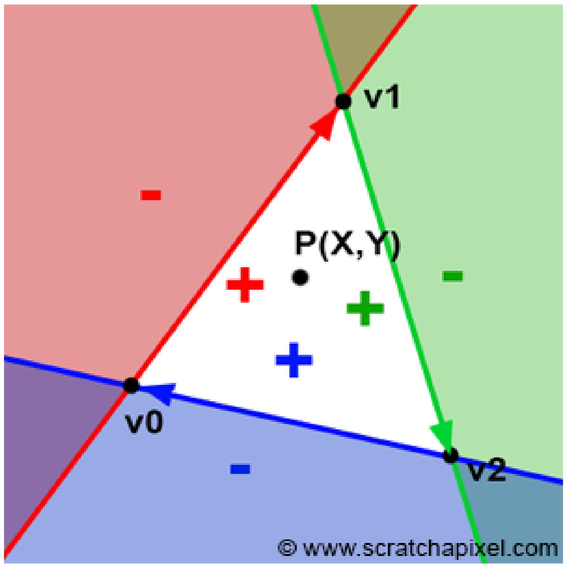

Figure 4: Cross product between triangle edges and a point determines if the point is inside the triangle [10].

Algorithm 1: Rasterization

```
Input: $s$, $\hat{d}$, $p$, $\mathcal{F}$, $\hat{u}$
Output: True if view to $p$ is not blocked.
$n_d \leftarrow (\mathcal{F} - s) \cdot \hat{u}$
$t \leftarrow \frac{n_d}{\hat{d} \cdot \hat{u}}$
for $f_i \in \mathcal{F}$
do
    if $t_i > t_p$ or $t_i < 0$ or $f_i$ is $p$ then
        continue
    $p_i \leftarrow s + d * t_i$
    $e_i \leftarrow [f_i[1] - f_i[0], f_i[2] - f_i[1], f_i[0] - f_i[2]]$
    $E_{01}(p_i) \leftarrow (e_i[0] \times (p_i - f_i[0]) \cdot \hat{u}_i$
    $E_{12}(p_i) \leftarrow (e_i[1] \times (p_i - f_i[1]) \cdot \hat{u}_i$
    $E_{20}(p_i) \leftarrow (e_i[2] \times (p_i - f_i[2]) \cdot \hat{u}_i$
    if $all(E) > 0$ then
        return False
end
return True
```

After computing all visible points for all samples $s \in S_i$, the sample g that views the most original visible points gets stored in the set $\mathcal{G}$. Every sample configuration set $S_i \in \mathcal{S}$ collects at least object point p_i that generated S_i, so the viewpoint generation algorithm complexity is $O(n)$ where n is the number of geometric points in the object P.

2.3 Finding a Minimum set of Base Positions

Each viewpoint represents a pose for the mobile robot's end-effector, and the end-effector pose requires an arm base position to start from. Inverse kinematics solve the manipulator joint angles between the base and end-effector goal pose. However, finding an inverse kinematic solution is very costly and many solutions for base positions exist. We simplify the inverse kinematics problem by restricting base positions to behind the viewpoint, and within an incidence angle, minimum and maximum distance. Figure 5 shows the simplified feasible base space as open cones with maximum distance of the robot arm length ℓ.

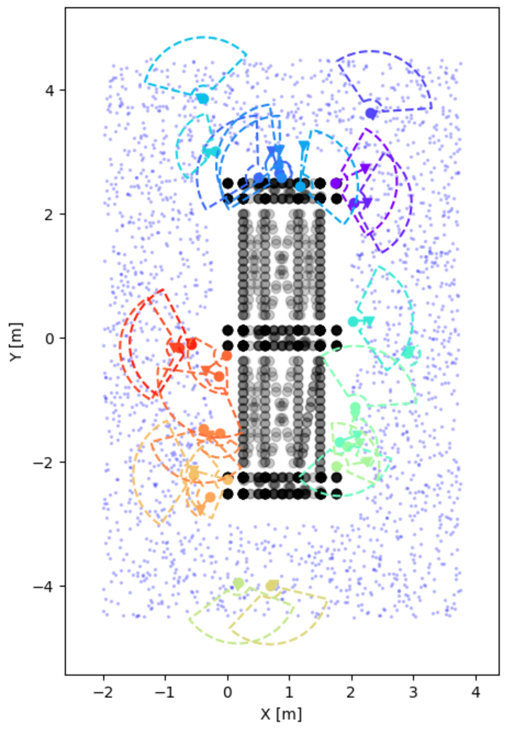

Figure 5: The object P, represented as black markers, surrounded by the free workspace $\mathcal{W}_f$. Viewpoints $\circ$ and corresponding base positions $\bigtriangledown$ drawn as same color and robot arm radius drawn as a circle with radius ℓ.

We propose an minimum set of base positions can be found using a complete hierarchial cluster. Hierarchical clustering combines data points based on their distance computed from a linkage function [8]. Many different types of linkage functions exist. The linkage that most applies to a robot arm reaching through space is the complete linkage. To understand how the complete linkage works, suppose a cluster u contains $u[0], u[1], \ldots, u[|u|-1]$ observations, and cluster v contains observations $v[0], v[1], \ldots, v[|v|-1]$. A *complete* linkage calculates the distance between two clusters $d(u, v)$ as the furthest distance between two elements in either cluster.

$$d(u, v) = \max(dist(u[i], v[j]))$$

We use a probabilistic road map to represent the mobile robot's free workspace $\mathcal{W}_f$ and calculate distances between viewpoints. If a viewpoint pair share a common sample spot within their feasible regions, then the distance between these two viewpoints is from that sample spot to the viewpoint position. Else if a viewpoint pair does not share a common sample, then the distance between the viewpoints is the path length between their nearest free space sample.

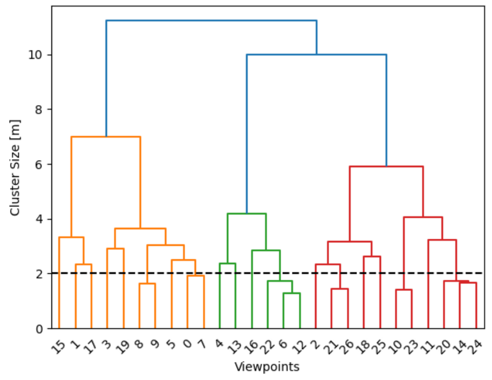

Figure 6: Dendrogram tree of complete heiarchial viewpoint clustering. The dashed line represents the robot arm reachable diameter 2ℓ.

A dendrogram in Figure 6 shows the clusters merging together at increasing distances. Branches are individual clusters and nodes represent when clusters accumulate another viewpoint. The y-axis represents the complete cluster size. The dashed line at distance 2ℓ represents the maximum robot arm radius. The number of branches at this level is the number of base positions that can reach all of the viewpoints. Nodes above the 2ℓ level require traversing a path between viewpoints.

2.4 Algorithm Summary

The complete algorithm returns a near optimal set of viewpoints $\mathcal{G}$ that completely cover an object of P and a minimum set of mobile base positions $\mathcal{B}$ that can reach those viewpoints. The algorithm is summarized below.

Algorithm 2: Mobile Manipulator Coverage Planner

```
Data: Polyhedron object $P$, Robot Model{radius $r$, arm length $\ell$}, vis. constraints{$\tau, dmin, dmax, fov$}
Result: a set of near optimal viewpoint poses $\mathcal{G}$ and minimum set of base poses $\mathcal{B}$
1  $\mathcal{X} \leftarrow \forall p_i \in P$
2  $\mathcal{W}_f \leftarrow PRM(\mathcal{W} - P, r)$
3  while $|\mathcal{X}| > 0$ do
4      $p \leftarrow rand(\mathcal{X})$
5      $S_i \leftarrow \mathcal{O}(p, \mathcal{W}_f, r, \ell, \tau, dmin, dmax)$
6      $g \leftarrow OPT(S_i, \mathcal{X})$
7      $\mathcal{G} \leftarrow g$
8      $\mathcal{X}.remove(\mathcal{V}(g; \tau, dmin, dmax, fov))$
9  end
10 $\mathcal{B} \leftarrow complete\_clustering(\mathcal{G}, \mathcal{W}_f)$
11 return $\mathcal{G}, \mathcal{B}$
```

3 Contribution

We propose a complete coverage path planning algorithm that utilizes a-priori knowledge of the triangular mesh representation of the structure to be inspected. The contribution of our work is two-fold: first, we propose a probabilistically complete random viewpoint generation approach that respects the constraints of our robotics system and sensor, and second, we present an efficient algorithm that utilized the decoupled behaviour of our redundant, non-homogeneous system such that it leverages existing open-source pipelines without much modification. The pipeline is implemented in simulation with 3D model of Tank and the effectiveness of the proposed autonomous approach is demonstrated.

4 Conclusion

This paper details a coverage planning algorithm suitable for a mobile robot with an extended arm. The planner uses an object model to randomly sample a minimal set of feasible viewpoints that completely see the object. Hierarchial clustering maps the viewpoints to a reachable base location the mobile robot moves to reach a set of viewpoints.

References

[1] Ercan Acar, Howie Choset, and Ji Yeong Lee. Sensor-based coverage with extended range detectors. *Robotics, IEEE Transactions on*, 22:189 – 198, 03 2006.

[2] Héctor Gonz alez Banós and Jean-Claude Latombe. Planning robot motions for range-image acquisition and automatic 3d model construction. *Integrated Planning for Autonomous Agent Architectures*, 1998.

[3] Andreas Bircher, Kostas Alexis, Michael Burri, Philipp Oettershagen, Sammy Omariand Thomas Mantel, and Roland Siegwart. Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. *IEEE International Conference on Robotics and Automation*, 2015.

[4] Howie Choset. Coverage for robotics - a survey of recent results. *Ann. Math. Artif. Intell.*, 31:113–126, 10 2001.

[5] Tim Danner and Lydia E. Kavraki. Randomized planning for short inspection paths. *IEEE International Conference on Robotics and Automation*, 2000.

[6] Brendan Englot and Hover Franz. Sampling-based coverage path planning for inspection of complex structures. *International Conference on Automated Planning and Scheduling*, 2012.

[7] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[8] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms, 2011.

[9] Atsushi Sakai, Daniel Ingram, Joseph Dinius, Karan Chawla, Antonin Raffin, and Alexis Paques. Pythonrobotics: a python code collection of robotics algorithms, 2018.

[10] Scratchapixel. Rasterization: a practical implementation (the rasterization stage).