

Coverage Path Planning in Large-scale Multi-floor Urban Environments with Applications to Autonomous Road Sweeping

Daniel Engelsons, Mattias Tiger and Fredrik Heintz

Conference paper

Cite this conference paper as:

Engelsons, D., Tiger, M., Heintz, F. Coverage Path Planning in Large-scale Multi-floor Urban Environments with Applications to Autonomous Road Sweeping, In (eds), 2022 International Conference on Robotics and Automation (ICRA), : Institute of Electrical and Electronics Engineers (IEEE); 2022, pp. 3328-3334. ISBN: 9781728196817

DOI: <https://doi.org/10.1109/ICRA46639.2022.9811941>

Copyright: Institute of Electrical and Electronics Engineers (IEEE)

<http://www.ieee.org/>

©2022 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

The self-archived postprint version of this conference paper is available at Linköping University Institutional Repository (DiVA):

<https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-188562>

Coverage Path Planning in Large-scale Multi-floor Urban Environments with Applications to Autonomous Road Sweeping

Daniel Engelsons¹ and Mattias Tiger¹ and Fredrik Heintz¹

Abstract—Coverage Path Planning is the work horse of contemporary service task automation, powering autonomous floor cleaning robots and lawn mowers in households and office sites. While steady progress has been made on indoor cleaning and outdoor mowing, these environments are small and with simple geometry compared to general urban environments such as city parking garages, highway bridges or city crossings. To pave the way for autonomous road sweeping robots to operate in such difficult and complex environments, a benchmark suite with three large-scale 3D environments representative of this task is presented. On this benchmark we evaluate a new Coverage Path Planning method in comparison with previous well performing algorithms, and demonstrate state-of-the-art performance of the proposed method. Part of the success, for all evaluated algorithms, is the usage of automated domain adaptation by in-the-loop parameter optimization using Bayesian Optimization. Apart from improving the performance, tedious and bias-prone manual tuning is made obsolete, which makes the evaluation more robust and the results even stronger.

I. INTRODUCTION

Enterprise and household robots routinely perform simple service tasks, with a demand for better performance, less maintenance and a wider set of tasks growing steadily [1]. Many of the most well-recognized automated service tasks today such as *lawn mowing*, *vacuuming* and *mopping* are powered by algorithms performing *Coverage Path Planning* (CPP) [2]. The CPP task is to generate a path that covers a map while avoiding obstacles. Steady progress has been made both regarding faster planners, which generates better (e.g. short) paths, and wider applicability [3]. CPP can be divided into two types, the *Sensor Coverage Problem* and the *Footprint Coverage Problem* [3]. Examples of the former is *3D exploration* [4], where a sensor has to be placed at sufficient poses as to fully capture a map. Examples of the latter is *floor cleaning* [5], where it is the robot, or its actuator, that has to be placed to cover the entire map. The Footprint Coverage Problem type of CPP is the focus here.

The CPP problem can be solved using A* search but it is intractable to solve to optimum for even small toy problems [6]. The reason for this is that CPP is NP-hard [7] and therefore necessitate approximate or heuristic algorithms for

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and by grants from the National Graduate School in Computer Science (CUGS), Sweden, the Excellence Center at Linköping-Lund for Information Technology (ELLIIT), the TAILOR Project funded by EU Horizon 2020 research and innovation programme GA No 952215, and Knut and Alice Wallenberg Foundation (KAW 2019.0350).

¹Daniel Engelsons, Mattias Tiger and Fredrik Heintz are with the Department of Computer and Information Science, Linköping University, Sweden. mattias.tiger@liu.se, fredrik.heintz@liu.se

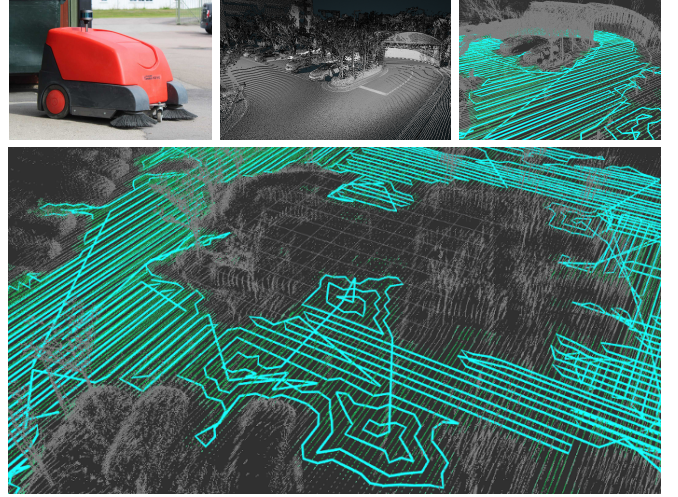


Fig. 1

Two-storey Garage point cloud. Coverage path, our approach. Gray points are inaccessible and green points are traversable. **Top left:** Concept example of a road sweeping robot.

solving CPP problems in practice. Algorithms have been developed that are tailored to specific application contexts, leveraging the specific circumstances such as flat surfaces on single floors, rigid shapes and small scales, which permit efficient use of grid representations. These are common characteristics of most indoor environment and outdoor lawns.

In this paper we consider offline CPP in the context of real-world *urban environments*. Such outdoor environments are characterized by a combination of large scales and complex surface geometry, with challenging aspects including non-flat surfaces, being partially cluttered and possibly multi-floor. A natural division between distinct rooms or floors are lacking, in contrast with general indoor environments studied in most of the literature [2], [3]. The surface geometry also differs from the characteristic regular straight lines and rectangular rooms of the indoor setting [3], and of the non-cluttered, rigid or smooth geometry of the outdoor setting in agriculture, such as croplands [8], [9], and customary lawns [10].

Real 3D environments are captured using LIDAR or 3D-sensors as point clouds. Further discretization can be made, e.g. using Octomap [11], but there are several benefits with working directly on a point cloud. Good coverage can be achieved while avoiding an intractably high voxel resolution, given a fairly even point distribution over surfaces and that the point density is kept down to manageable levels. Furthermore, while planar or voxel-based surface approximations

can be efficient representations, motion planning on such approximations may lead to unfeasible plans, unless care is taken. For example, a plan that seemed to be the shortest of all explored alternatives can be impossible to follow, lead to unsafe execution or be worse (e.g. take longer to follow) than other plans which seem worse during planning [12].

The task investigated in this paper is that of CPP for autonomous road sweeping (Figure 1), where the input is a point cloud and the output is a collision free path with highest possible coverage while minimizing path length and total rotation. The main contributions of this paper are:

- A benchmark suite¹ consisting of three large-scale urban environments as point clouds, with and without labeled points. They span a variety of multi-floor and complex geometry, representative of outdoor road sweeping tasks under difficult and varying real-world conditions.
- Evaluation of prominent CPP algorithms on the urban outdoor CPP task.
- A proposed CPP algorithm that out-performs state-of-the-art CPP algorithms at this task.
- Parameter optimization of CPP algorithms for automatic domain adaptation, for evaluation and as part of an enhanced offline meta-CPP algorithm.

The outline is as follows. In Section II the problem is formally summarized, in Section III related work is discussed and relevant background material is presented in IV. Section V describe how 3D point clouds are analyzed to determine the sets of points traversable and coverable by the robot. The proposed CPP algorithm is detailed in Section VI, and the benchmark suite including, three 3D point clouds representative for the urban road sweeping task, is presented in VII. Section VIII contains the evaluation of CPP algorithms on the benchmark and Section IX concludes the paper.

II. PROBLEM FORMULATION

Consider a robot with geometry $R \subset \mathbb{R}^3$ that can travel over a surface with a maximum height difference of h_{\max}^R . Given a bounded 3D world $\mathcal{W} \subset \mathbb{R}^3$ and a starting point x_S , the set $\mathcal{W}_{\text{cov}} \subseteq \mathcal{W}$ contains all surface points coverable by the robot. A surface point p is coverable if there is a point $x \in \mathcal{W}$ for which the robot placed at x cover p with its geometry R , and if x is reachable from x_S without collisions.

The Coverage Path Planning (CPP) problem is to find a path P of points $x \in P$ such that every point $p \in \mathcal{W}_{\text{cov}}$ must have been inside R at least once when placing R at every position in P . The task here is to determine \mathcal{W}_{cov} and subsequent a path starting in x_S that let the robot cover all coverable points $x \in \mathcal{W}_{\text{cov}}$ while minimizing the path cost.

III. RELATED WORK

Coverage path planning is still an open problem in robotics [13], with some of the issues being planning efficiency, path optimality, handling dynamic obstacles and dealing with complex environments. These issues are considered some of the major challenges in robotics, and CPP particularly

challenging in complex and large-scale environments. For large-scale environments, offline CPP algorithms are the most common due to limitations on onboard hardware [13].

The development of robots for floor cleaning and road sweeping has a long history [14] and so does the development of algorithms for coverage path planning for robotics [2]. Although the mechatronic and control development of road sweeping robots keep advancing [15], very little prior work [16] exists on applying CPP to autonomous road sweeping and none that we can find in a realistic setting. One reason for the past dominance of CPP in the indoor setting [2] can be due to the computational issues of scaling up current methods [13], not least grid- and cell-based methods [17], to the large scales of urban outdoor areas. Outdoor urban environments pose many challenges even to standard path planning [12], and thus also to coverage path planning.

A recent survey [3] of CPP methods for indoor floor cleaning conclude after extensive empirical evaluations that Boustrophedon motion is the most prominent performer on covering single rooms. Multi-room single-floor apartments are separated into individual rooms covered in isolation. In [17] they note that a major weakness of BA* [18], which apply Boustrophedon motion locally such as for each room, is the long paths between the locally covered regions.

The general poor performance of grid-based methods on large scale environments can be alleviated by rectangular decomposition as a pre-processing step [17], making the proposed method suitable for floor cleaning tasks in large scale environments such as a gym, library, warehouse or airports. They also consider cluttered indoor environments, e.g. gym equipment and library book shelves. The environment is, apart from these aspects, similar to the general characteristics in the indoor literature with a single rectangular room, simple and straight borders, and a planar and single floor.

The geometry of non-planar surfaces can be taken into account to make the coverage path more energy efficient [19], for example by driving orthogonal to the slope. The drive direction can also affect the actual coverage achievable in cases where elevation is projected down onto a grid and solved using grid-based CPP methods [9]. By solving CPP in 3D instead, such as directly on the point cloud, such projection based issues are eliminated. Energy efficiency can be taken into consideration directly in the motion planning and it is straight forward to plan over multiple floors [12].

IV. BACKGROUND

Coverage path planning algorithms relevant for understanding the proposed algorithm are described here, followed by the parameter optimization technique utilized in this work.

The CPP algorithms BA* [18] and Inward Spiral [20] are representative of a family of methods which are well performing in a variety of settings. The proposed algorithm (VI) combine ideas from these. An overview of the algorithms are presented here, see the references for further details.

A. CPP algorithm: BA*

BA* [18] is based on Boustrophedon motions [21], which are a kind of zigzag motion. The idea is to cover a local

¹github.com/danneengelsson/urbanroadsweeper

regions with a zigzag pattern, then find the shortest path to the next uncovered position and repeat the behavior until completeness. The algorithm consists of the following steps:

- 1) Starting at starting point x_s , with main direction ϕ , cover the local area using the Boustrophedon motion (BM) algorithm until a critical point is reached and no further Boustrophedon motion is possible.
- 2) Use a backtracking list to find the next starting point.
- 3) Use A* [22] to plan a collision free path to next point.
- 4) Shorten the path using the A*SPT [18] algorithm.
- 5) Follow the path and go to step 1 to cover a new area.

These steps are repeated until Step 2 can no longer find a new uncovered starting point. Note that ϕ is a design-parameter.

B. CPP algorithm: Inward Spiral

Inward Spiral [20] apply counter-clockwise motion from initial point x_s . The algorithm consists of the following steps:

- 1) Starting at starting point, clean area in an inward spiral motion until a dead zone is reached.
- 2) Find closest uncovered accessible point with BFS [22].
- 3) Find shortest path using A* to the closest uncovered point, the next starting point, and go back to Step 1.

This cycle repeats until the area has been covered.

C. Parameter Optimization

Bayesian Optimization (BO) [23] is a gradient-free global optimization method powered by probabilistic inference, where the uncertainty of the objective function is modelled across the input range. It is a sequential process: The next input to evaluate is the most likely input candidate to produce the lowest expected loss given all previously investigated inputs so far. BO is much more sample efficient than other types of optimization methods. It is well suited for situations where it is costly to try a new input value, which makes it important to try out as few as possible while still having a reasonably high probability of finding a global minimum.

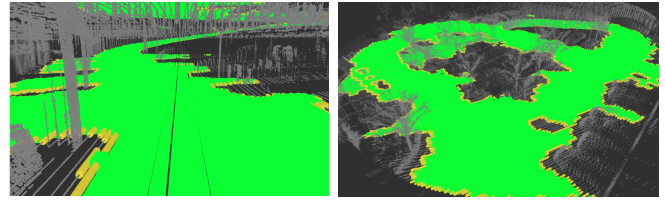
HyperOpt [24] is a widely used parameter optimization library implementing Bayesian Optimization. It is most often used for parameter tuning in machine learning, but the applicability of BO is widespread [23] and in this work it is used as an automatic method to adapt the CPP algorithms to the specific domain. Tedious and ad hoc hand tuning is replaced with a systematic and automatic modern method.

V. TERRAIN ASSESSMENT

To navigate a robot in an environment, knowledge about traversable positions is required. As in [12] we chose to keep the terrain representation *continuous* and move between 3D positions. Since we do both motion planning and covering, coverable positions also needs to be known. See Figure 2 for examples. To guarantee that traversable and coverable positions are drivable they are restricted to classified points.

The purpose of the terrain assessment is to classify points in a given point cloud. Points are classified as:

- **Inaccessible** - Unreachable or can not overlap with robot geometry due to collision or lack of information.



(a) View of floor 1.

(b) View of floor 2.

Fig. 2

Terrain Assessment of the Two-storey Garage point cloud. Green points represent traversable areas, yellow are non-traversable but coverable and grey points are inaccessible.

- **Traversable** - Can be visited by the robot without collision with an obstacle.
- **Coverable** - Can be covered by the range of the robot from at least one traversable point.

Our method is based on robot specific properties such as robot height, maximum navigable step height, covering range and robot breadth. The covering range and the breadth are assumed to be the same and defined by a radius from current position of the robot.

- 1) **Floor segmentation** - Divide the point cloud into different floors by splitting the points into vertically stacked layers and finding heights with the largest amount of points as in [25].
- 2) **Cell segmentation** - For each floor separately. Create a Digital Elevation Model (DEM) of the floor by splitting the point cloud into cells and finding the elevation using the method in [25]. It looks for free space between two points in a cell larger than the robot height. If found, the elevation is set to the lowest point.
- 3) **Cell classification** - A Breadth First Search is then used to find the biggest connected area of coverable cells. The DEM is used to prevent steps where the height difference between cells is too big.
- 4) **Point Classification** - Classify points with these steps:
 - a) Set all points that are not in the main coverable area as *Inaccessible*
 - b) Set all points in the main coverable area that are more than a robot radius away from an *Inaccessible* point as *Traversable*.
 - c) Set all points that are within a robot radius from a *Traversable* point as *Coverable*.

This method does not guarantee that every traversable point is accessible from the main coverable area. Step 4b) makes narrow passages nontraversable which separates clusters of points from the main coverable area and consequently, make 100% coverage unreachable.

The CPP methods in Section IV are designed for operation on a grid or cell-network. To work on a point cloud it is necessary to define how valid steps, from one traversable point to another, is done. In this work we use 8-connectivity on the virtual ground plane in the 3D point cloud manifold. Each CPP algorithm has a step size λ_{CPP} and a visited distance-threshold r_{visited} which controls valid expansions of

the algorithms. Given a traversable point, a neighbour point is found in either direction by selecting the closest traversable point λ_{CPP} away in the specific direction. The neighbour is already visited if a previously expanded traversable point is within a r_{visited} distance of the neighbour. Since r_{visited} is not the actual cleaning radius of the robot, it controls the allowance of revisiting cleaned spots.

Algorithm 1: Sample Based BA* & Inward Spiral.

Data: Starting position x_s . $P = \emptyset$. $S_{\text{TSP}} = \emptyset$.
Result: Path P to cover the area.

```

1  $S_{\text{BA}^*} \leftarrow \text{Sample BA}^*$  (Algorithm 2)
2  $S_{\text{IS}} \leftarrow \text{Sample Inward Spiral}$  (Algorithm 3)
3  $S \leftarrow S_{\text{BA}^*} \cup S_{\text{IS}}$ 
4  $G \leftarrow \text{Empty fully-connected graph}$ 
5 for path  $P_i$  in  $S$  do
6   | Add start and end point of  $P_i$  as nodes in  $G$ .
7 end
8  $P_{\text{TSP}} \leftarrow \text{Cheapest route to visit all nodes in } G$ 
9  $x_{\text{curr}} \leftarrow x_s$ 
10 for  $x_i$  in  $P_{\text{TSP}}$  do
11   | if  $p_i \neq x_{\text{curr}}$  then
12     | if  $x_i$  is a start point then
13       |  $P_i \leftarrow \text{Path in } S \text{ with } x_i \text{ as start point}$ 
14     | else
15       |  $P_i \leftarrow \text{Reversed path in } S \text{ with } x_i \text{ as end point}$ 
16     | end
17     |  $S_{\text{TSP}} \leftarrow S_{\text{TSP}} \cup P_i$ 
18     |  $x_{\text{curr}} \leftarrow \text{Last point in } P_i$ 
19   | end
20 end
21  $x_{\text{curr}} \leftarrow x_s$ 
22 for path  $P_i$  in  $S_{\text{TSP}}$  do
23   |  $P_A^* \leftarrow \text{Shortest path from } x_{\text{curr}} \text{ to the start of } P_i$ .
24   |  $P \leftarrow P \oplus P_A^* \oplus P_i$ 
25   |  $x_{\text{curr}} \leftarrow \text{Last point in } P_i$ 
26 end
27 return  $P$ 
```

VI. PROPOSED METHOD
(SAMPLED BA* & INWARD SPIRAL)

- 1) Initialise an empty list of detached path segments S .
- 2) **PART I:** Line 5-12: For each of the 4 different main directions ϕ , plan a local coverage path using the BA* algorithm [18] starting from a traversable edge point of a randomly sampled uncovered area. Stop covering when the euclidean distance to the next starting point in the backtracking list (IV-A) exceeds d_{max}^1 .
- 3) If none of these 4 paths had a $\frac{\text{cost}}{\text{coverage}}$ that were lower than C_{min}^1 , set the points covered by the path with the biggest coverage as *explored* (line 14). Otherwise, choose the path with the lowest cost per coverage. Add it to S and set its covered points as *explored* and *covered* (line 16-17).
- 4) Repeat Step 2-3 until E of the points are *explored*.
- 5) **PART II:** Start covering using the Inward Spiral algorithm [20] starting from a traversable edge point of a randomly sampled uncovered area. Stop covering when the distance to the next starting point (IV-B)

Algorithm 2: PART I: Sample BA*

Data: $S = \emptyset$.
Result: Set of paths S covering local regions.

```

1 while exploration goal  $E$  has not been reached do
2   |  $x' = \text{random position in uncovered area}$ 
3   |  $x_r = \text{closest edge point to } x' \text{ given by BFS}$ 
4   |  $S_{\text{BA}^*} = \emptyset$ 
5   | for angle  $\phi \in [0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}]$  do
6     |  $x_s \leftarrow x_r$ 
7     | do
8       |  $P_{\text{BA}^*} \leftarrow P_{\text{BA}^*} \cup [x_s, \dots, p_e] = \text{BA}^*(x_s, \phi)$ 
9       |  $x_s \leftarrow \text{new starting point in backtracking list}$ 
10      | while  $\|p_e - x_s\| < d_{\text{max}}^1$ ;
11      |  $S_{\text{BA}^*} = S_{\text{BA}^*} \cup P_{\text{BA}^*}$ 
12    | end
13    | if  $C_{\text{min}}^1 < \max_{P_{\text{BA}^*} \in S_{\text{BA}^*}} \frac{\text{cost}}{\text{coverage}}$  then
14      | Mark  $P_{\text{BA}^*} \in S_{\text{BA}^*}$  with max coverage as explored
15    | else
16      |  $P_{\text{best}} \leftarrow P_{\text{BA}^*} \in S_{\text{BA}^*}$  with lowest  $\frac{\text{cost}}{\text{coverage}}$ 
17      | Mark  $P_{\text{best}}$  as explored and covered
18      |  $S = S \cup P_{\text{best}}$ 
19    | end
20 end
21 return  $S$ 
```

Algorithm 3: PART II: Sample Inward Spiral

Data: $S = \emptyset$.
Result: Set of paths S covering local regions.

```

1 while coverage  $\neq C$  and exploration  $\neq 1$  do
2   |  $x' = \text{random position in uncovered area}$ 
3   |  $x_s = \text{closest edge point to } x' \text{ given by BFS}$ 
4   | do
5     |  $P_{\text{Spiral}} \leftarrow P_{\text{Spiral}} \cup [x_s, \dots, p_e] = \text{Inward-Spiral}(x_s)$ 
6     |  $x_s \leftarrow \text{new starting point using BFS from } p_e$ 
7     | while  $\|p_e - x_s\| < d_{\text{max}}^2$ ;
8     | Mark  $P_{\text{Spiral}}$  as explored
9     | if  $(\frac{\text{cost}}{\text{coverage}} \text{ of } P_{\text{Spiral}}) < C_{\text{min}}^2$  then
10      |  $S \leftarrow S \cup P_{\text{Spiral}}$ 
11      | Mark  $P_{\text{Spiral}}$  as covered
12    | end
13 end
14 return  $S$ 
```

exceeds d_{max}^2 . Set the points that were covered by the path as *explored*. If the cost per coverage is lower than C_{min}^2 , add the path to S and set its covered points as *covered*.

- 6) Repeat Step 5 until C of the points are *covered* or all points have been *explored*.
- 7) **PART III:** Add the start and end positions of the paths in S as nodes to a fully connected graph G . All nodes are connected with edges with weights set to

- $w = 0$, if the edge is between the start and end node of the same path in S .
- Otherwise, distance based according to equation

$$w = w_{\text{offset}} + \|s_A(x, y) - s_B(x, y)\| + K\|s_A(z) - s_B(z)\|$$

where s_A and s_B are the positions of the two

nodes, w_{offset} is a large number and $K \geq 1$.

The purpose of w_{offset} is to make sure that the traveling salesman algorithm in the following step always chooses to connect corresponding start and end nodes. Since the environment could have multiple floors, extra weight K is added to difference in height, to avoid potential movement between floors.

- 8) Solve the TSP problem [26] of the cheapest visitation order, P_{TSP} , to visit all nodes in G .
- 9) Walk through every node in P_{TSP} and create an ordered list of paths S_{TSP} . For every node s_i ,
 - If s_i is start node, add corresponding path to S_{TSP} .
 - If s_i is an end node, add the corresponding path, but reversed, in S to S_{TSP} .

- 10) Create a continuous path P by following the paths in S_{TSP} . Connect them with obstacle free smooth paths.

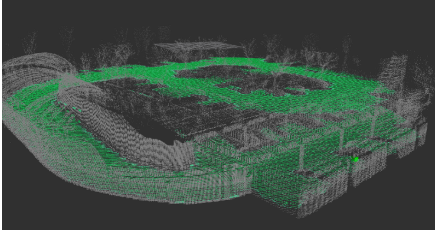
The algorithm parameters are listed in Table I.

TABLE I
Sampled BA* & Inward Spiral parameters.

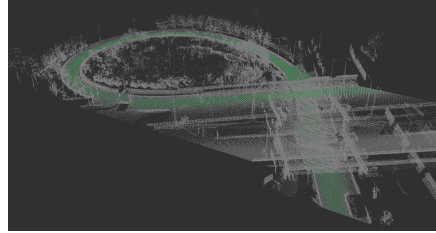
Parameter	Description
E	Exploration goal for PART I
C	Total coverage goal
d_{max}^1	Max distance to next starting point in PART I
d_{max}^2	Max distance to next starting point in PART II
C_{min}^1	Min coverage-normalized cost in PART I
C_{min}^2	Min coverage-normalized cost in PART II
λ_{CPP}	Step size
r_{visited}	Visited threshold

VII. AUTONOMOUS ROAD SWEEPING

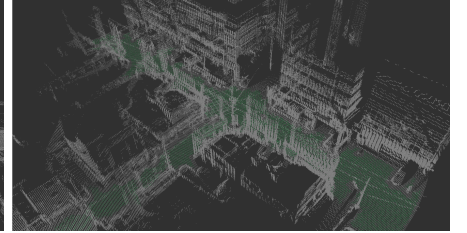
Road sweeping is a challenging task due to a complex environment, where the environment is somewhat unbounded and may have rough terrain [16]. A particular difficulty of contemporary CPP algorithms is that of large-scale environments [13], even with simple geometry. We consider the challenging task of a road sweeping robot in an urban setting, where offline CPP is sufficient but where CPP is



(a) Two-storey Garage

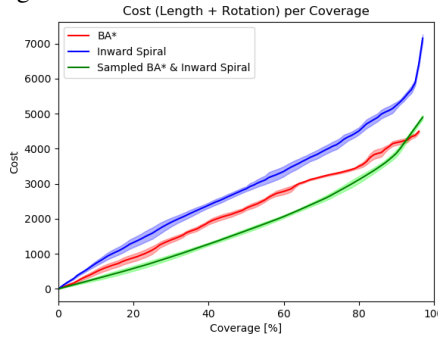


(b) Highway Bridge

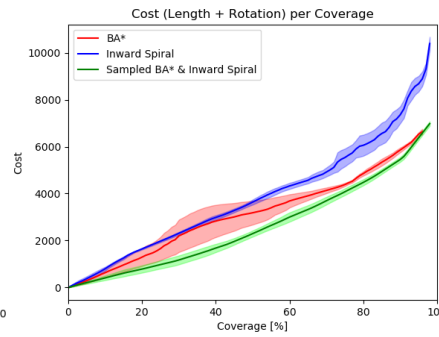


(c) City Crossing

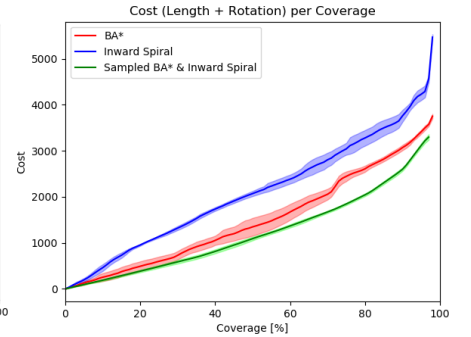
Fig. 3



(a) Two-storey Garage



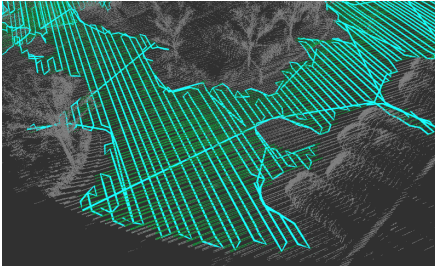
(b) Highway Bridge



(c) City Crossing

Fig. 4

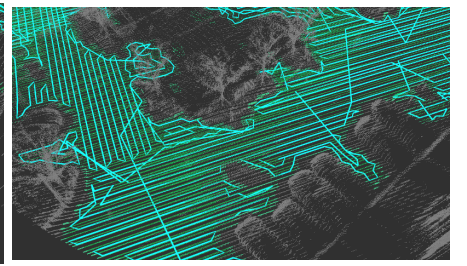
Generalization results to new starting points. Mean and 95% confidence interval over 10 random starting locations.



(a) BA*



(b) Inward Spiral



(c) Sampled BA* & Inwards Spiral

Fig. 5

Coverage paths on the Garage environment.

performed on a large-scale map with complex geometry. The benchmark suite consists of three environments (VII-A) which are realistically represented as point clouds, collected from real urban environments. Part of the suite is also a scenario description (VII-B) which include specifications of a road sweeping robot, providing the necessary details to perform terrain assessment (V) and to evaluate CPP algorithms on the benchmark environments.

A. 3D Environments

The three urban environments (Figure 3) consist of 3D point clouds that have been carefully selected from the large Complex Urban Dataset [27]. These environments represent different difficult, but typical, urban outdoor scenes.

The *Two-storey Garage* environment consists of a parking lot with an underground garage right below. The point cloud is a subset of *urban05* [27]. The main challenges are complex border geometry, multi-level aspect and the transition between levels. It contains 2.6 M points and has a total coverable area for the road sweeping robot of 1902 m². The *Highway Bridge* environment is a subset of *urban17*, contains 2.4 M points and has a total coverable area for the road sweeping robot of 2707 m². The main challenges are the very long curved drive-way together with its multi-level aspect. The *City Crossing* environment is a subset of *urban02*, contains 3.2 M points and has a total coverable area for the road sweeping robot of 1296 m². The main challenge is the clutter and complex surface/border geometry.

B. Scenario

A 3D point cloud of an urban environment is available for offline CPP. A road sweeping robot (e.g. Figure 1) of dimension length = 1 m, breadth = 0.75 m, height = 0.8 m is tasked with cleaning the ground surface of the target environment. The actuators are most effective when driving straight and quickly lose effectiveness when turning wide.

Given a point cloud and a starting point, traversable and coverable points are classified and collected using terrain assessment (V). A maximum a posteriori (with respect to cost) path is found by optimizing a specified CPP algorithm over its parameters using Bayesian Optimization (IV-C) such as HyperOpt [24], based on prior distributions over the parameters. The parameter optimization perform domain adaptation, and a much better path for the given domain and task is found than if an arbitrary parameter configuration is used that has not been optimized for the problem instance. This is a meta algorithm for domain adaptive offline CPP. The resulting path is executed by a motion planner capable of collision avoidance [12], [28]. To deal with new stationary obstacles: plan repair [29] to next free node in CPP plan.

VIII. EVALUATION

The CPP algorithm parameters are optimized using HyperOpt [24] based on a single starting position per environment. Parameters are sampled from the distributions in Table II. We ran 100 evaluations of HyperOpt for each algorithm, for each environment, from the same start position. Table III lists the

results, where the proposed method is consistently significantly better than the other two methods in cost and rotation across all environments. The parameter optimization makes a noticeable impact to the performance of all algorithms over using the prior mean, $\mathbb{E}[\text{Prior}]$. To evaluate the robustness of the domain adaptation to other starting positions, i.e. measure the generalization of the MAP parameters, another 10 start points are randomized for respective environment and solved using all three algorithms. The results are presented in Figure 4, where the proposed algorithm still outperforms BA* and Inward Spiral, apart from for the highest degree of coverage on Two-storey Garage. Examples are shown in Figure 5.

TABLE II

CPP parameters optimized using HyperOpt on start position.

BA*				
Parameter	Garage	Bridge	Crossing	Prior
λ_{CPP}	0.634 m	0.557 m	0.524 m	$\mathcal{U}(0.5, 1)$
r_{visited}	0.473 m	0.453 m	0.404 m	$\mathcal{U}(0.25, 0.5)$
ϕ	4.65 rad	5.34 rad	4.70 rad	$\mathcal{U}(0, 2\pi)$
Inwards Spiral				
Parameter	Garage	Bridge	Crossing	Prior
λ_{CPP}	0.628 m	0.737 m	0.665 m	$\mathcal{U}(0.5, 1)$
r_{visited}	0.499 m	0.483 m	0.500 m	$\mathcal{U}(0.25, 0.5)$
Sampled BA* & Inwards Spiral				
Parameter	Garage	Bridge	Crossing	Prior
E	0.865	0.940	0.863	$\mathcal{U}(0.75, 0.95)$
d_{max}^1	4.13 m	4.49 m	1.69 m	$\mathcal{U}(1, 5)$
d_{max}^2	6.94 m	7.06 m	4.48 m	$\mathcal{U}(4, 10)$
C_{min}^1	8238	12773	6488	*
C_{min}^2	13645	25989	8141	**
λ_{CPP}	0.548 m	0.619 m	0.554 m	$\mathcal{U}(0.5, 1)$
r_{visited}	0.373 m	0.389 m	0.258 m	$\mathcal{U}(0.25, 0.5)$

* Based on coverable area A : $\mathcal{U}(2.63A, 5.26A)$

** Based on coverable area A : $\mathcal{U}(5.26A, 10.52A)$

TABLE III

Result of meta-CPP domain adapted solution.

Env.	Algorithm	Cost ($\mathbb{E}[\text{Prior}]$)	Length	Rotation
Garage	Sampled BA*...	4077 (4422)	2999 m	1078 rad
	BA*	4238 (5270)	2896 m	1342 rad
	Inwards Spiral	5787 (9351)	3101 m	2686 rad
Bridge	Sampled BA*...	5797 (6519)	4643 m	1154 rad
	BA*	6385 (8545)	4562 m	1823 rad
	Inwards Spiral	7213 (16637)	4440 m	2773 rad
Crossing	Sampled BA*...	3001 (3248)	2186 m	815 rad
	BA*	3262 (4058)	2249 m	1013 rad
	Inwards Spiral	4054 (6647)	2239 m	1815 rad

IX. CONCLUSIONS

A realistic benchmark suite for advancing the CPP capabilities of service robot tasks in urban environments is presented, together with a novel state-of-the-art algorithm which combines the strengths of three other prominent CPP algorithms. A rigid domain adaptation methodology for CPP is proposed, and the generalization of the domain adaptation is demonstrated in the evaluation. Using Bayesian optimization we optimize the non-exact CPP algorithms for the best possible path they can generate. The next step towards field robotics is to evaluating the proposed algorithm and domain adaptation approach in practice onboard a real robot sweeper.

REFERENCES

- [1] C. B. Frey and M. A. Osborne, "The future of employment: How susceptible are jobs to computerisation?" *Technological forecasting and social change*, vol. 114, pp. 254–280, 2017.
- [2] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.
- [3] R. Bormann, F. Jordan, J. Hampp, and M. Hägele, "Indoor coverage path planning: Survey, implementation, analysis," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1718–1725.
- [4] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1500–1507, 2020.
- [5] R. Bormann, J. Hampp, and M. Hägele, "New brooms sweep clean: an autonomous robotic cleaning assistant for professional office cleaning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4470–4477.
- [6] S. Dogru and L. Marques, "A*-based solution to the coverage path planning problem," in *Iberian Robotics conference*. Springer, 2017, pp. 240–248.
- [7] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry*, vol. 17, no. 1–2, pp. 25–50, 2000.
- [8] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *Journal of field robotics*, vol. 26, no. 8, pp. 651–668, 2009.
- [9] I. A. Hameed, A. la Cour-Harbo, and O. L. Osen, "Side-to-side 3d coverage path planning approach for agricultural robots to minimize skip/overlap areas between swaths," *Robotics and Autonomous Systems*, vol. 76, pp. 36–45, 2016.
- [10] N. Einecke, J. Deigmöller, K. Muro, and M. Franzius, "Boundary wire mapping on autonomous lawn mowers," in *Field and Service Robotics*. Springer, 2018, pp. 351–365.
- [11] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, 04 2013.
- [12] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart, "Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments," *Journal of Field Robotics*, vol. 34, no. 5, pp. 940–984, 2017.
- [13] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms," *IEEE Access*, 2021.
- [14] E. Prassler, A. Ritter, C. Schaeffer, and P. Fiorini, "A short history of cleaning robots," *Autonomous Robots*, vol. 9, no. 3, pp. 211–226, 2000.
- [15] M. M. Rayguru, R. E. Mohan, R. Parween, L. Yi, A. V. Le, and S. Roy, "An output feedback based robust saturated controller design for pavement sweeping self-reconfigurable robot," *IEEE/ASME Transactions on Mechatronics*, 2021.
- [16] M.-S. Chang, J.-H. Chou, and C.-M. Wu, "Design and implementation of a novel outdoor road-cleaning robot," *Advanced Robotics*, vol. 24, no. 1–2, pp. 85–101, 2010.
- [17] X. Miao, J. Lee, and B.-Y. Kang, "Scalable coverage path planning for cleaning robots using rectangular map decomposition on large environments," *IEEE Access*, vol. 6, pp. 38 200–38 215, 2018.
- [18] H. H. Viet, V.-H. Dang, M. N. U. Laskar, and T. Chung, "Ba*: an online complete coverage algorithm for cleaning robots," *Applied intelligence*, vol. 39, no. 2, pp. 217–235, 2013.
- [19] S. Dogru and L. Marques, "Towards fully autonomous energy efficient coverage path planning for autonomous mobile robots on 3d terrain," in *2015 European Conference on Mobile Robots (ECMR)*. IEEE, 2015, pp. 1–6.
- [20] H. Zhang, W. Hong, and M. Chen, "A path planning strategy for intelligent sweeping robots," in *International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2019, pp. 11–15.
- [21] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and service robotics*. Springer, 1998, pp. 203–209.
- [22] S. J. Russell and P. Norvig, *Artificial intelligence : a modern approach*. Boston: Pearson, 2020.
- [23] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.
- [24] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International conference on machine learning*. PMLR, 2013, pp. 115–123.
- [25] V. Sakenas, O. Kosuchinas, M. Pfingsthorn, and A. Birk, "Extraction of semantic floor plans from 3d point cloud maps," in *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*, 2007, pp. 1–6.
- [26] F. Goulart, "Python tsp solver," 2021. [Online]. Available: <https://github.com/fillipe-gsm/python-tsp>
- [27] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *International Journal of Robotics Research*, vol. 38, no. 6, pp. 642–657, 2019.
- [28] O. Andersson, O. Ljungqvist, M. Tiger, D. Axehill, and F. Heintz, "Receding-horizon lattice-based motion planning with dynamic obstacle avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4467–4474.
- [29] M. Wzorek, J. Kvarnström, and P. Doherty, "Choosing replanning strategies for unmanned aircraft systems," in *International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 2, no. 3, 2010.