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

Daniel Engelsons[1] and Mattias Tiger[1] and Fredrik Heintz[1]

***Abstract*— Coverage Path Planning is the work horse of contemporary service task automation, powering autonomous floor cleaning robots and lawn mower 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 supposed 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 moving*, *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 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.

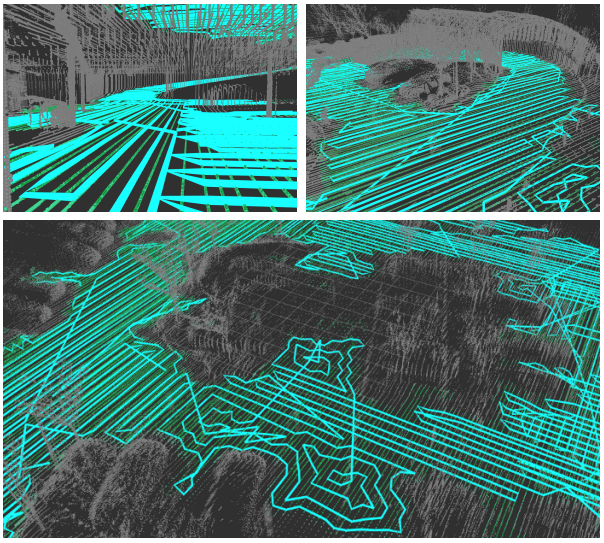


Fig. 1
Two-storey Garage point cloud. Coverage path, our approach. Gray points are inaccessible and green points are traversable.

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 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, for example 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 that the point cloud distribution is fairly even 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. A plan that seemed to be the shortest of all explored alternatives can be impossible to follow, lead

[1]Danil 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

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 suite[1] 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^R_{max}. Given a bounded 3D world $\mathcal{W} \in \mathbb{R}^3$ and a starting point x_S, the set $\mathcal{W}_{cov} \subseteq \mathcal{W}$ contain all surface points that the are coverable by the robot. A surface point p is coverable by the robot 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 for the robot without collisions.

The Coverage Path Planning (CPP) problem is to find a path P of states $x \in P$ such that every point $p \in \mathcal{W}_{cov}$ 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 $x \in \mathcal{W}_{cov}$ 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].

[1]Code and benchmark suite will be made available upon paper acceptance.

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 environments in an urban setting pose many challenges even to standard path planning [12], and consequently so to coverage path planning.

A recent survey [3] of CCP 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, with each room being solved for a coverage path in isolation. In [17] they note that a prominent weakness of BA* [18], which apply Boustrophedon motion locally such as in 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* [20] and Inward Spiral [21] 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, for more details see the references.

A. CPP algorithm: BA*

BA* [20] is based on Boustrephedon motions [18], which are a kind of zigzag motion. The idea is to cover a local

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 following steps:

1) 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 the next starting point.
4) Shorten the path using the A*SPT [20] 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.

B. CPP algorithm: Inward Spiral

Inward Spiral [21] works by counter-clockwise motion. The algoritm consists of the following steps:

1) Clean area in an inward spiral motion until a dead zone
2) Find closest uncovered accessible point using Breadth-first search [22].
3) Find shortest path using A* to the closest uncovered 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 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.

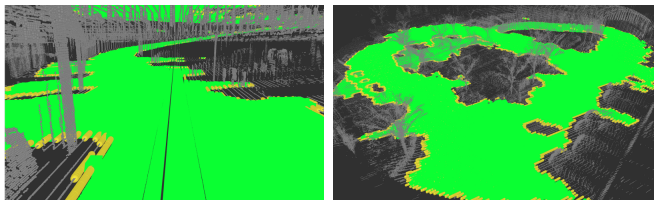
HyperOpt [24] is a widely used parameter optimization library implementing Bayesian Optimization. It is most often used for parameter tuning in machine learning, but the applicability of BO is widespread [23] and in this work it is used as an automatic method to adapt the CPP algorithms to the specific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

V. Terrain Assessment

To navigate a robot in a 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 are in addition to motion planning are doing covering, coverable positions also needs to be known. See Figure 2 for examples. To always guarantee that traversable and coverable positions are on a drivable surface they are restricted to classified points in the point cloud.

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

- **Inaccessible** - Can not overlap with robot geometry.



(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** - Could be covered by the range of the robot after visiting every 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 slitting the point cloud into cells and finding the elevation using method proposed in [25]. It looks for free space between two points in the cell that are bigger 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.

A. CPP on a Point Cloud

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 point cloud manifold. Each CPP algorithm have a step size λ_{CPP} and a visited distance-threshold $r_{visited}$ which control 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.

Algorithm 1: Sample Based BA* & Inward Spiral.

Data: Starting position p_s. $W = \emptyset$
Result: Path W to cover the area.
$S_{\text{BA*}} \leftarrow$ Sample BA* (Algorithm 2)
$S_{\text{IS}} \leftarrow$ Sample Inward Spiral (Algorithm 3)
$S \leftarrow S_{\text{BA*}} \cup S_{\text{IS}}$
$T \leftarrow$ Empty graph tree
for path W_i in S **do**
  Add start and end point of W_i as nodes in T.
end
$W_{\text{TSP}} \leftarrow$ Cheapest route to visit all nodes in T
$p_{curr} \leftarrow p_s$
$S_{\text{TSP}} \leftarrow \emptyset$
for node p_i in W_{TSP} **do**
  if $p_i = p_{curr}$ **then**
    continue
  end
  if p_i is a start node **then**
    $W_i \leftarrow$ Path in S with p_i as start node
  else
    $W_i \leftarrow$ Reversed path in S with p_i as end node
  end
  S_{TSP}.push_back(W_i)
  $p_{curr} \leftarrow$ Last point in W_i
end
$p_{curr} \leftarrow p_s$
for path W_i in S_{TSP} **do**
  $W_{A*} \leftarrow$ Shortest path from p_{curr} to the start point of W_i.
  $W \leftarrow W \cup W_{A*}, W_i$
  $p_{curr} \leftarrow$ Last point in W_i
end
return W

Algorithm 2: PART I: Sample BA*

Data: $S = \emptyset$
Result: Set of paths S covering local regions.
while Exploration E^1 has not been reached **do**
  s_r = random position in uncovered area
  p_r = closest border point to s_r given by BFS
  $S_{BA*} = \emptyset$
  for angle $\phi \in [0, 1, ..., N_\phi] \cdot 2\pi/N_\phi$ **do**
    $p_s \leftarrow p_r$
    while *distance top*$_s < d^1_{max}$ **do**
      $W_{BA*} \leftarrow$ BA* at p_s with north $= \phi$
      $p_s \leftarrow$ new starting point in backtracking list
    end
    $S_{BA*} = S_{BA*} \cup W_{BA*}$
  end
  $S_{BA*_{accepted}} \leftarrow W_{BA*}$with $\frac{\text{cost}}{\text{coverage}} > C^1_{min}$
  if $S_{BA*_{accepted}} = \emptyset$ **then**
    $W_{best} \leftarrow W_{BA*} \in S_{BA*}$with max coverage
  else
    $W_{best} \leftarrow W_{BA*} \in S_{BA*}$with lowest $\frac{cost}{coverage}$
    $S = S \cup W_{best}$
  end
  $E^1 \leftarrow$ covered points byW_{best}
end
return S

Algorithm 3: PART II: Sample Inward Spiral

Data: $S \leftarrow \emptyset$
Result: Set of paths S covering local regions.
while Goal coverage C has not been reached **do**
  s_r = random position in uncovered area
  p_r = closest border point to s_r given by BFS
  $p_s \leftarrow p_r$
  while *distance top*$_s < d^2_{max}$ **do**
    $W_{Spiral} \leftarrow$ Inward Spiral at p_s
    $p_s \leftarrow$ new starting point using BFS
  end
  if $\frac{cost}{coverage}$ *of* $W_{Spiral} > C^2_{min}$ **then**
    $S \leftarrow S \cup W_{Spiral}$
  end
end
return S

VI. Proposed Method (Sampled BA* & Inward Spiral)

We propose a new CPP method, *Sampled BA* & Inward Spiral*, which combines the strengths of BA* [20] and Inward Spiral [21], with inspiration taken from Sampling-Based Sweep Planning [26]. The idea is to minimise the cost, defined as length + rotation, by covering open areas with BA* and more complex areas with Inward Spiral. This is done by covering segments of the area with BA* and Inward Spiral starting from random sampled points and then connect these segments with a TSP solver. Sampled BA* & Inward Spiral (Algorithm 1) consists of the following steps:

1) Initialise a list of segments S, covered points P_{cov} and explored points P_{exp}.
2) For $N_\phi = 4$ different angles ϕ_i, define $north$ as the angle ϕ_i and start covering using the BA* algorithm [20] starting from a *Traversable* border point of a randomly sampled uncovered area. Stop covering when the distance to the next starting point in the backtracking list exceeds d^1_{max}.
3) If no path had a cost per coverage that were lower than C^1_{min}, choose the path with the biggest coverage and add all covered points by the chosen path to P_{exp}. Otherwise, choose the path with the lowest cost per coverage. Add it to S and add points that were covered by it to P_{cov} and P_{exp}.
4) Repeat Step 2-3 until the exploration percent, given by the size of P_{exp}, reaches E^1.
5) Start covering using the Inward Spiral algorithm [21] starting from a *Traversable* border point of a randomly sampled uncovered area. Stop covering when the distance to the next starting point exceeds d^2_{max}. If cost per coverage is lower than C^2_{min}, add path to S.
6) Repeat the Step 5-6 until P_{cov} gives coverage $> C$.
7) Repeat Step 5-6 until the exploration percent, given by the size of P_{exp}, reaches E^1.

TABLE I
Sampled BA* & Inward Spiral parameters.

Parameter	Description
E^1	Exploration goal for PART I
C	Total coverage goal
d^1_{max}	Max distance to next starting point in PART I
d^2_{max}	Max distance to next starting point in PART II
C^{min}_1	Min cost (length + rotation) per coverage in PART I
C^{min}_2	Min cost (length + rotation) per coverage in PART II
λ_{CPP}	Step size
$r_{visited}$	Visited threshold

8) Build up a graph tree, T, from S. Only the start and end positions of the paths in S are added as nodes to T. All nodes are connected with edges with weights
   - $w = 0$, if the edge is between the start and end node of the same path in S.
   - Otherwise, distance based according to equation

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

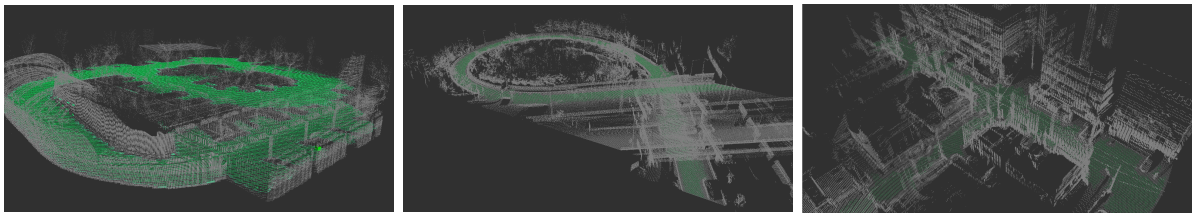
   where s_A and s_B are the position of the two nodes, w_{offset} is a large number and $K \geq 1$.

   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.
9) Solve the TSP problem [27] of the cheapest visitation order, W_{TSP}, to visit all nodes in T.
10) Walk through every node in W_{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 a end node, add the corresponding path, but reversed, in S to S_{TSP}.
11) Create a continuous path W by following the paths in S_{TSP}. Connect them with obstacle free smooth paths.

The algorithm parameters are listed in Table I.

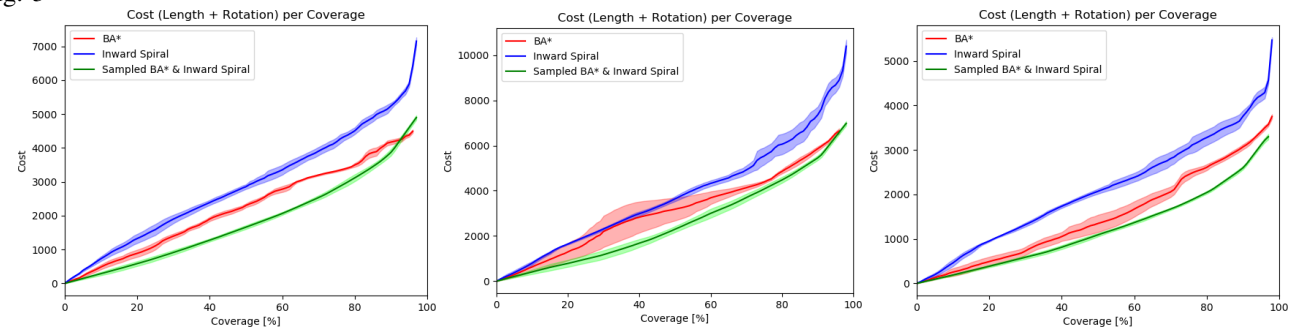
VII. Autonomous Road Sweeping

Road sweeping is a challenging task due to the a complex environment, where the environment is somewhat unbounded



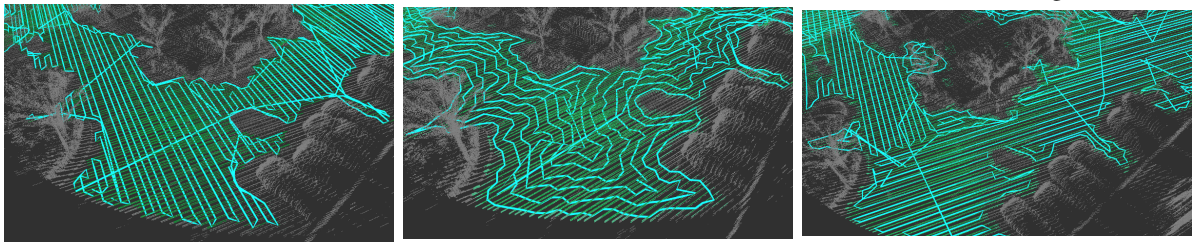
(a) Two-storey Garage (b) Highway Bridge (c) City Crossing

Fig. 3



(a) Two-storey Garage (b) Highway Bridge (c) City Crossing

Fig. 4
Performance results on the benchmark suite. Mean and 95% confidence interval over 10 random staring locations.



(a) BA* (b) Inward Spiral (c) Sampled BA*

Fig. 5
Coverage paths on the Garage environment.

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 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) consists of 3D point clouds that have been carefully selected from the large Complex Urban Dataset [28]. These environments represent different difficult, but typical, urban outdoor scenes.

The *Two-storey Garage* environment consists of a parking lot and an underground garage right below the parking lot. The point cloud is a subset of *urban05* [28]. The main challenges are the complex border geometry, the multi-level aspect and the transition between levels. It contains 2.6M points and has a total traversable area for the road sweeping robot of 1678m^2. The *Highway Bridge* environment is a subset of *urban17*, contains 2.3M points and has a total traversable area for the road sweeping robot of 2637m^2. The *City Crossing* environment is a subset of *urban02*, contains 3.2M points and has a total traversable area for the road sweeping robot of 1296m^2.

B. Scenario

A road sweeping robot of dimension length $= 1m$, breadth $= 0.75m$, height $= 0.8m$ is tasked with cleaning the ground surface in an urban environment. The actuators are most effective when driving straight and quickly lose effectiveness when turning wide.

A 3D point cloud of the target environment is available for offline CPP. 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], [29].

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. 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}	0.634m	0.557m	0.524m	$\mathcal{U}(0.5, 1)$
$r_{visited}$	0.473m	0.453m	0.404m	$\mathcal{U}(0.25, 0.5)$
ϕ	4.65 rad	5.34rad	4.70 rad	$\mathcal{U}(0, 2\pi)$
Inwards Spiral				
Parameter	**Garage**	**Bridge**	**Crossing**	**Prior**
λ_{CPP}	0.628m	0.737m	0.665	$\mathcal{U}(0.5, 1)$
$r_{visited}$	0.499m	0.483	0.500	$\mathcal{U}(0.25, 0.5)$
Sampled BA* & Inwards Spiral				
Parameter	**Garage**	**Bridge**	**Crossing**	**Prior**
E^1	0.865	0.940	0.863	$\mathcal{U}(075, 0.95)$
d^1_{max}	4.13m	4.49m	1.69m	$\mathcal{U}(1, 5)$
d^2_{max}	6.94m	7.06m	4.48m	$\mathcal{U}(4, 10)$
C^1_{min}	8238	12773	6488	*
C^2_{min}	13645	25989	8141	**
λ_{CPP}	0.548m	0.619m	0.554	$\mathcal{U}(0.5, 1)$
$r_{visited}$	0.373	0.389m	0.258	$\mathcal{U}(0.25, 0.5)$

* Based on coverable area A: $\mathcal{U}(0.165A, 0.33A)$
** Based on coverable area A: $\mathcal{U}(0.33A, 0.66A)$

TABLE III
Result of meta-CPP domain adapted solution.

Environment	**Algorithm**	**Cost**	**Length**	**Rotation**
Garage	Sampled BA*	**4077**	2999	**1078**
	BA*	4238	**2896**	1342
	Inwards Spiral	5787	3101	2686
Bridge	Sampled BA*	**5797**	4643	**1154**
	BA*	6385	4562	1823
	Inwards Spiral	7213	**4440**	2773
Crossing	Sampled BA*	**3001**	**2186**	**815**
	BA*	3262	2249	1013
	Inwards Spiral	4054	2239	1815

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 combine the strengths of three other algorithms. A rigid domain adaptation methodology for CCP is proposed, and the generalization of the domain adaptation is demonstrated in the evaluation. Using Bayesian optimization we optimize the non-exact CCP 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. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and service robotics*. Springer, 1998, pp. 203–209.

[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. 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.

[21] 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.

[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] B. Englot and F. S. Hover, "Sampling-based sweep planning to exploit local planarity in the inspection of complex 3d structures," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 4456–4463.

[27] F. Goulart, "Python tsp solver," 2021. [Online]. Available: https://github.com/fillipe-gsm/python-tsp

[28] 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.

[29] 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.