2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

# Indoor Coverage Path Planning: Survey, Implementation, Analysis

Richard Bormann, Florian Jordan, Joshua Hampp, Martin Hägele

*Abstract*— **Coverage Path Planning (CPP) describes the process of generating robot trajectories that fully cover an area or volume. Applications are, amongst many others, mobile cleaning robots, lawn mowing robots or harvesting machines in agriculture. Many approaches and facets of this problem have been discussed in literature but despite the availability of several surveys on the topic there is little work on quantitative assessment and comparison of different coverage path planning algorithms. This paper analyzes six popular off-line coverage path planning methods, applicable to previously recorded maps, in the setting of indoor coverage path planning on room-sized units. The implemented algorithms are thoroughly compared on a large dataset of over 550 rooms with and without furniture.**

## I. Introduction

In this work we analyze the performance of several coverage path planning algorithms. Coverage path planning (CPP) is the computation of a path that fully covers a given map. It is relevant, for example, to mobile cleaning robots for floor cleaning [1], [2], lawnmowers [3], or autonomously driving agricultural machines [4]. Our motivation to engage in this comparison is the application of professional office cleaning robots [5], which have to fulfill several purposes. On the one hand, such a robot shall be capable of exhaustively cleaning larger areas, e.g. with a wet cleaning machine. On the other hand, the robot must be capable of quickly inspecting larger areas for the occurrence of waste bins to clear or local floor pollutions to clean. Because of its complexity, the coverage path planning problem is usually not computed on complete maps but rather on individual parts of a map [5], [6]. In our previous work, we already set up a pipeline for the automatic segmentation of a map into single room units and the computation of an efficient visiting order of these rooms in the sense of a Traveling Salesman Problem [5], [7]. The work in this paper is concerned with the generation of efficient coverage paths within the resulting room-level units with methods that can handle both robot footprint-based path coverage as well as sensor-based coverage with real sensors of wider but limited range.

The paper is intended to supplement recent surveys on coverage path planning [6], [8] with experimental validation in common building maps and provides open source implementations to 6 different algorithms. Coverage planning can have various requirements such as finding the shortest path, avoiding collisions, trading maximal coverage with time budgets, or just generating attractive and intuitive movement patterns. Moreover, the problem has several variants, e.g. the Art Gallery Problem, in which a minimal amount of stationary sensors has to be placed in an environment such that complete observability is provided, or the Watchman Tour Problem, which tries to maximize observability with a minimum costs tour through the environment [9]. All of these problems are NP-hard and hence obtaining optimal solutions is only feasible for very limited problem domains. Beyond this barrier, appropriate heuristics have to be employed. This paper provides implementations and a thorough analysis of a broad range of coverage path planning approaches including one exact cell decomposition method, three cell grid-based planners, one contour line-driven approach and one variant of the art gallery problem. Literature distinguishes between online algorithms (sensor-based coverage algorithms) and offline algorithms, which require a complete map for computing the coverage path [8]. The work in this paper focuses on offline algorithms that can utilize a readily available floor plan of the environment. In particular, the main contributions of this paper are:

1) an overview on indoor CPP for room-sized units;
2) the paper is accompanied by open source implementations of six coverage planning algorithms [1];
3) with the Contour Line-based method we propose a new approach to coverage path planning;
4) we provide a large collection of over 550 room maps with and without furniture;
5) a qualitative and quantitative comparison of these algorithms.

This project has received funding from the German Federal Ministry of Economic Affairs and Energy (BMWi) through grant 01MT16002A (BakeR).

The authors are with the Robot and Assistive Systems Department, Fraunhofer IPA, 70569 Stuttgart, Germany. `<first name>.<last name>@ipa.fraunhofer.de`

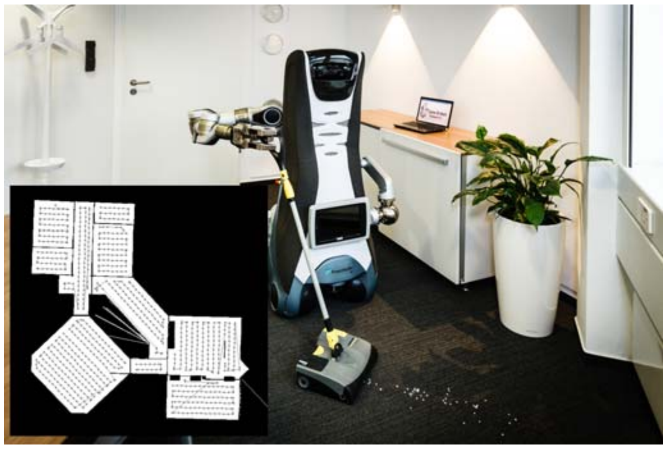
Fig. 1. A cleaning robot needs systematic inspection or cleaning patterns for working efficiently.

[1]All six implementations are available as ROS package from `http://wiki.ros.org/ipa_room_exploration`.

The remainder of this paper is organized as follows: Sec. II briefly summarizes several areas of related work before Sec. III provides a description of the six implemented methods of this paper, which become analyzed for their performance in Sec. IV.

## II. Related Work

Most of the relevant work in the field of coverage path planning has been summarized and discussed in the surveys of Choset [8] and Galceran and Carreras [6]. Here we provide a brief summary of the different kinds of relevant approaches to the Coverage Path Planning problem at room-level.

One group of methods is constituted by the classical exact cellular decompositions methods [10]–[12], the Morse-based cellular decomposition methods [13], as well as the Landmark-based cell decomposition algorithms [14], which divide the original map into smaller units that can be covered with a simple motion pattern. Whereas the classical exact cellular decompositions usually rely on polygonal structures and obstacles, this limitation is lifted with Morse-based decompositions. The popular Boustrophedon cell decomposition [11], [15] belongs to the cell decomposition methods and applies a simple back-and-forth motion inside the generated cells. The detection of critical cell decomposition points has been generalized by Wong and MacDonald [14] to any kind of topological landmarks. Rotating the sweep line or cells for optimal boustrophedon patterns has been proposed by Huang [16].

Another type of coverage algorithms are cell grid-based methods, i.e. methods that divide the map into a regular grid of cells and find a path that covers all of these cells. The Wavefront Algorithm [17] defines a starting and a goal cell and propagates a wavefront from the goal to the start. Cells are visited in equidistant level sets of these wave fronts before approaching the target further. The Spanning Tree method [18] decomposes the free space into mega cells and constructs a spanning tree that covers all of these cells. Inside the mega cells there are 4 smaller cells, than can be visited by traversing the spanning tree. Both methods guarantee coverage but yield quite erratic movement patterns. A biologically inspired method for covering a cell grid is the Neural Network-based coverage path planner [19] that considers all cells as neurons and decides for the next cell to visit by the activation state of the neighboring cells in the network.

The floor coverage or inspection task can also be considered as an Art Gallery Problem [20], or a Watchman Tour Problem. Arain et al. [9] provide a good overview on optimal and approximate methods in this domain and propose an new approach, which uses re-weighted convex relaxation in addition to the combinatorial integer linear programming method to scale up the size of the problems that can be tackled.

From the plethora of algorithms, we selected a diverse set of six approaches and implemented them for comparing their performance. The respective algorithms are explained in the following section.

## III. Methods

This section explains the six coverage path planning algorithms in detail which are evaluated in this paper. We assume that map data is available as a grid map with sufficient resolution, i.e. at most the size of the grid cells of the coverage planning algorithms. Furthermore, we require that the provided map is pre-processed by some room segmentation procedure which splits larger map areas into smaller segments at the size range of ordinary rooms. The coverage path planning methods are supposed to operate on these room-like working areas. For most of the implemented coverage path planning algorithms we normalize each room for its orientation before computing the coverage path.

### A. Preparation

Both pre-processing steps, map segmentation and room orientation normalization, are briefly outlined in the following paragraphs. Furthermore, we discuss the modeling of the coverage area.

*1) Map Segmentation:* We proposed a set of suitable map division algorithms in our earlier work [7]. However, even the best of these algorithms are not completely free of sporadic errors, and moreover, the evaluation of coverage planning algorithms within this paper shall not be biased by the use of a certain segmentation method. Hence, we decided to utilize the ground truth segmentations provided together with the maps from our map segmentation dataset [21]. After segmentation a grid map is created for each individual room for further processing. The optimal path along the set of segmented rooms can be found as the solution of a Traveling Salesman Problem as described in our earlier work [7].

*2) Room Orientation Normalization:* All of the implemented algorithms, except for the contour line-based coverage planning, operate on some kind of pose grid laid over the room area or a cell decomposition. The choice of orientation of such a grid can have significant influence on the performance of the coverage algorithm, specifically on the number of cell decompositions or the number of parallel tracks and time-intense spot turns [6], [12]. Aligning the grid with the major direction of the enclosing walls is a commonly applied heuristic [12], [16]. Specifically, we convert the grid map to an image with white regions representing the accessible room areas and dark regions at the walls and obstacles. Then the room contours are recovered with the Canny edge detection algorithm, followed by the Hough straight line detector from OpenCV [22]. This procedure fits a set of straight lines which approximate the enclosing walls of the room and larger obstacles. We begin with minimum required line length of 1 m and decrease this target value if too few lines are found. The orientations of all Hough lines are collected in a histogram with $10^\circ$ bins. The orientation correction for the room alignment is taken from the exact average rotation provided by the entries in the histogram's maximum bin.

*3) Coverage Area:* All implementations support the definition of the coverage area with an arbitrary quadrilateral which may be placed off the robot's center, or a robot-centered circular area. Consequently, the coverage area can model a circular footprint-based cleaning device such as a vacuum cleaner or a lawn mower, as well as the field of view (FOV) from a sensor mounted on a robot such as a camera facing the ground. The definition of coverage device affects the maximum grid cell size for the underlying coverage planning algorithm if complete coverage is desired. In case of a circular, robot-centered coverage device with radius $r_c$ we use a square grid of at most $l = \sqrt{2} \cdot r_c$ side length to fully cover the grid cell with the robot when located in its center. The upper left sketch in Fig. 2 illustrates this setting.

Similarly, we find a maximal incircle that fits into the arbitrary quadrilateral coverage area by constructing the skeleton or Voronoi graph [23], [24] inside the quadrilateral. The Voronoi graph departs at the corners $A, B, C, D$ of the quadrilateral along the angle bissectors, which meet in one point $M$ altogether for tangential quadrilaterals or pairwise in two points $E, F$ inside the quadrilateral (see Fig. 2). In the first case, $M$ is the circle center and its radius $r_q$ is the closest distance to the quadrilateral sides. In the second case, $E$ or $F$ is the circle center, depending which occurs with greater distance to the quadrilateral. The distance between circle center and quadrilateral sides can be easily obtained in a grid map via the distance transform. Again, the square grid for coverage planning can be at most $l = \sqrt{2} \cdot r_q$ in terms of the incircle radius $r_q$ if the coverage area shall be guaranteed to fully cover the grid cell from any perspective.

If such coverage guarantees are not necessary, e.g. because the coverage algorithm ensures trajectories along straight lines such that the coverage (in)circle touches all parts of a grid cell during the linear motion, the grid spacing can be increased to $l = 2r_c$.

In those cases when the coverage device's center does not coincide with the robot center we conduct coverage planning in two stages: first, the coverage path for the viewpoint center is determined (as it would be a robot center), and following, we compute suitable robot poses at the respective offsets to the viewpoint centers. Here the favorite viewing angle on a pose is given by the direction from the previous pose, yielding a smooth path without unnecessary movements. If the favored pose is not accessible in the map, another suitable pose is found on the circle around the target viewpoint.

After these pre-processing steps we can compute a coverage path on the individual room areas utilizing one of the following approaches.

### B. Boustrophedon Coverage Path Planning

The boustrophedon approach is an exact cellular decomposition method which divides the room area into smaller cells using a sweep line. Our implementation follows the formulation of Choset et al. [11], [15]. W.l.o.g. we assume that our room orientation normalization aligned the room with its longest dimension along the x-axis and hence a horizontal sweep line is shifted from the top to the bottom of the room map. Whenever the sweep line hits critical points, i.e. when one segment is separated into two by an obstacle, or when two separate segments merge together after an obstacle, the space is divided and merged accordingly yielding a set of cells that can be covered with a simple back-and-forth motion pattern (see Fig. 3). Each cell is analyzed again for its major orientation as described in Sec. III-A.2 to set up the back-and-forth motion pattern in a way that maximizes the length of single laps and minimizes the number of time-taking turns.

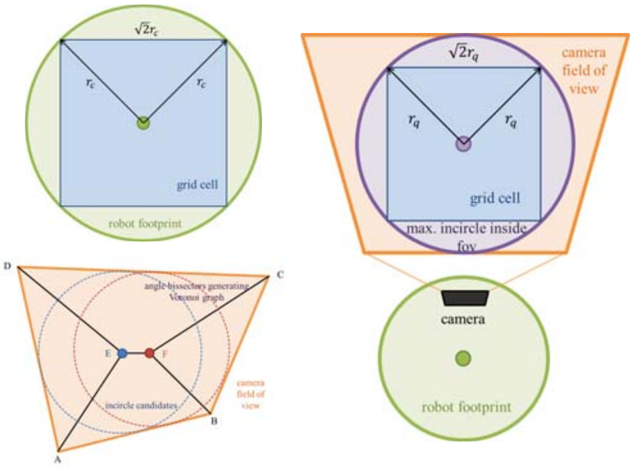

Fig. 2. Maximum grid size with a centered footprint coverage device (upper left) or a projective sensor (right). The lower left graphic explains how the incircle with maximum radius is found in the field of view.

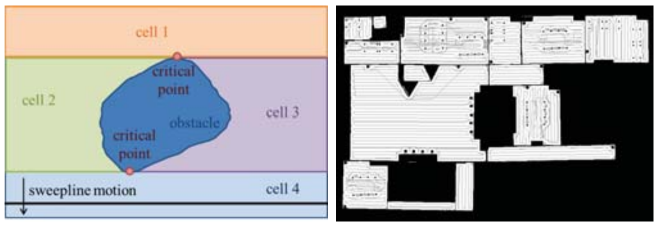

Fig. 3. Boustrophedon cell decomposition (left) and generation of simple movement patterns inside the cells (right).

### C. Grid-based Traveling Salesman Coverage Path Planning

The Grid-based Traveling Salesman Problem (TSP) planner decomposes the room map into a regularly spaced cell grid $\mathcal{G}$ with appropriate cell size $l$ according to Sec. III-A.3, where grid cells are only generated at accessible areas of the map or shifted into an accessible area with a maximum offset of $l$ from their original location. The solution to the coverage path planning problem is a path that visits all of these accessible grid cells. One possible path throughout the set of grid cells is the globally shortest traveling path. Consequently, solving the Traveling Salesman Problem on cell set $\mathcal{G}$ offers a valid solution to the original coverage path planning problem. The traveling distances between pairs of grid cells are approximated with the length of the shortest path connecting them as determined by the A$^*$ algorithm on the original room map. The TSP is solved exactly with the Concorde solver [25], or approximately with a genetic algorithm or with the simple yet fast nearest neighbor approach.

## D. Neural Network-based Coverage Path Planning

In contrast to the globally optimizing grid-based TSP planner the Neural Network-based Coverage Path Planner [19] optimizes the path along the same cell grid $\mathcal{G}$ in a semi-global fashion inspired by the activation dynamics of neurons. All grid cells are called neurons and they have a connecting edge to their eight direct neighbors. Dynamics of a neuron are modeled with a shunting equation that contains attractive terms from not yet visited neurons, repulsive terms for obstacles and walls, as well as neutral terms for visited cells. Activation is spread within the local 8-neighborhood. We refer to the original publication [19] for the complete formulae of the activation dynamics. The next neuron $p_n$ to visit is chosen in each step according to the activation state $x_j, j \in 1, 2, \ldots, 8$ of the 8 neighboring neurons and the change in moving direction of the robot $\Delta\theta_j$

$$p_n \Leftarrow x_{p_n} = \max\left\{x_j + c\left(1 - \frac{\Delta\theta_j}{\pi}\right)\right\}, \qquad (1)$$

where $c$ is a constant. The neighbor with highest score is chosen, even if it has been visited before. This behavior is necessary if the algorithm got stuck in a corner and then needs to follow the trace of activation leading to further unvisited neurons. After a movement the activations of all neurons are updated and the next movement is selected until all neurons have been visited.

## E. Grid-based Local Energy Minimization

The third grid-based coverage path planning algorithm is purely driven by local energy minimization within the current neighborhood. It has been proposed in our earlier work on cleaning robots [5]. Again, the room area is populated with the regular grid $\mathcal{G}$ that has a spacing of $l$. The robot starts at a point $p_0$ in a corner of the room. Each visited grid cell is appended to a list $L$ of processed locations. The next unseen target location is selected from the grid which minimizes the energy functional

$$E(p,n) = d_t(p,n) + d_r(p,n) + N(n) \qquad (2)$$

$$d_t(p,n) = \frac{\sqrt{(p_x - n_x)^2 + (p_y - n_y)^2}}{l} \qquad (3)$$

$$d_r(p,n) = \frac{\|p_\theta - n_\theta\|_{\text{angle}}}{\pi/2} \qquad (4)$$

$$N(n) = 4 - \sum_{k \in \text{Nb}_8(n)} \frac{|k \cap L|}{2} \qquad (5)$$

where $p = (x, y, \theta)$ denotes the current robot pose and $n$ is the coordinate of the potential next location. The translational distance $d_t$ is measured in units of $l$, the rotational distance $d_r$ in units of $\pi/2$. Function $N(n)$ represents half the number of not yet visited locations among the 8 neighbors $\text{Nb}_8(n)$ around $n$ and constitutes an attractive term for the robot path to stay next to already processed grid cells. This term results in a track following behavior. If there are unvisited neighbors in the 8-neighborhood of $p$, one of these is chosen as next robot pose. If all neighbors around $p$ have been visited already the potential next poses $n$ are drawn from the set of unvisited grid cells $\mathcal{G}\backslash L$. The algorithm never actively visits grid cells twice (but might pass close to them) and terminates when all grid cells have been visited. Similar to the neural network planner, this procedure would also be applicable online to incomplete maps and it can directly react to moving obstacles if the cell grid is updated accordingly.

## F. Contour Line-based Coverage Path Planning

The contour line-based method is the only one in our selection that does not require any kind of cell grid or map orientation normalization. It can be computed directly on the room map. First, it finds the points in the map with maximum distance to obstacles (center points) by computing the skeleton of a Voronoi graph. Next, the Voronoi graph is divided into individual centers at critical points. Critical points occur at local minima on the Voronoi graph, i.e. at narrow passages between two larger areas. The direct connection between the critical point and the closest obstacle points is chosen to divide the space to cover. To this end, the division of the map is similar to a Voronoi cell decomposition [7], [26].

Inside each Voronoi cell a distance transform is computed, which represents the distance of each accessible map cell to the closest obstacle or the cell border, yielding equidistant contour lines to obstacles within the cell. Given the coverage width $w_c = 2 \cdot r_c$ of the robot, contour levels are now chosen as $c_i = r_c + i \cdot w_c$ such that the largest contour line $c_{max} < d_{max}$ does not exceed the largest distance in the Voronoi cell $d_{max}$. All points on the same contour level sets are now connected minimizing costs, including distance between points, direction change, and if the point was already cleaned/visited similar to Eq. 2. Likewise do the individual level trails become connected between each other eventually.

## G. Convex Sensor Placement Coverage Path Planning

A completely different approach to the coverage planning problem is represented by the art gallery problem, which is originally concerned with the optimal placement of as few sensors or guards as possible to ensure complete observation of the target area. However, the optimal placement of a sensor mounted on a mobile robot directly translates the art gallery problem into a coverage path planning problem. Arain et al. [9] propose the Convex Sensor Placement algorithm which first selects a minimum set of candidate sensing configurations from which all cells in the map are visible. Afterward, the shortest connecting path of these observation positions is found as the solution of a Traveling Salesman Problem. For the first step, an integer linear program is formulated and solved to find the poses that in sum have the least sensing cost while covering the whole space. Assuming uniform costs this results in choosing the fewest number of sensing poses. An initial approximative solution to the integer linear program is found much faster with an iterative re-weighted convex relaxation via $\ell_1$-minimization. For details of the algorithm we refer to the original publication [9].

## IV. Evaluation

In this section the algorithms of Sec. III are evaluated under various conditions on more than 550 different room maps to reveal the individual strengths and weaknesses. We utilize the 20 maps from our previous work on segmenting maps into individual rooms [7], [21] and obtain the room maps for this work from the ground truth segmentations for avoiding bias with one of the possible segmentation approaches. This yields a total of over 550 rooms that are available either with or without furniture. The focus of this work is on off-line algorithm performance, i.e. the full map is already known in advance before computing the coverage path. The simulated robot is allowed to replace inaccessible commanded trajectory points with suitable accessible points within a radius of 1.5 times the grid cell size $l$ during trajectory execution. Only if no suitable pose can be found in the vicinity, the commanded pose is abandoned. All experiments were simulated on a single core of a mobile I7 6800 HQ processor with 32 GB RAM. The evaluation is broadly divided into a footprint coverage problem (Sec. IV-A) and a sensor coverage problem (Sec. IV-B). Both approaches differ in their placement and range of coverage device.

### A. Footprint Coverage Problem

In this setting the coverage device is mounted at the robot center and typically has limited extent, e.g. the robot radius. This problem is very common for floor cleaning robots or robotic lawnmowers. All six path planning algorithms have been applied to all 550 rooms in the configuration of a circular footprint with coverage radius $r_c = 0.3$ m. For quantitative comparison the generated paths were analyzed with respect to several measures. First, the algorithm computation time was recorded. Furthermore, the average path length per room as well as the average accumulated absolute orientation changes of the robot base $\sum_i |\Delta\theta_i|$ were taken. The average traveling time is composed of path length driven at a speed of 0.3 m/s and rotations at a rotary speed of 0.52 rad/s. Together with the percentage of actually covered ground floor by driving the computed path, these values are summarized in Table I, which distinguishes between the cases with and without furniture. Table I also provides a subjective *appeal* parameter that tries to assess the subjective attractiveness of the generated paths to a cleaning professional. The parameter is the better the higher. It contains the parallelism of the path against walls and previously traveled parts of the path as a positive contribution (parallelism is in range [0,1] each) and is reduced by the following four terms:

- times of crossing the old path divided by room area,
- path length divided by room area,
- accumulated rotations divided by $\pi$ and room area, and
- return time into the vicinity of already visited parts of the robot trajectory measured in percentage of trajectory completion (measure for local track completion).

The design of this appeal number has been influenced by the discussion with professional cleaning service providers and cleaning machine dealers who consistently agreed that a client would be more happy with a robot that drives a suboptimal but easily predictive path compared to an optimal but hardly understandable path. Parallelism is a positive contribution in this sense, crossing the path a negative. Longer paths and many rotations are also considered critical.

The results of Tab. I indeed indicate that each algorithm has specific strengths and weaknesses. With respect to computation time the heuristic approaches Boustrophedon, Neural Network, Grid Local Energy, and Contour Line always achieve a high performance taking less than 4 s per room. In contrast, the optimal solvers Grid TSP and Convex SPP are quite slow with up to 41.6 s per room on average and few times going beyond 30 min with rising problem complexity, i.e. room size. Vice versa, Grid TSP and Convex SPP are always amongst the methods which achieve the shortest paths. The Neural Network planner takes erratic path lengths, which are the longest in comparison, because this planner can get temporarily trapped in cycles of already visited nodes.

The Boustrophedon algorithm always has a very low number of rotations, since it was specifically designed for few rotations. The Grid Local Energy planner behaves similarly since its energy term specifically minimizes rotations (esp. visible in Tab. II). The number of rotations increases for furnished maps for the heuristic approaches because the regular movement patterns are interrupted more frequently by obstacles. With respect to absolute traveling time Boustrophedon, Contour Line, and Grid Local Energy are the best choices in rooms with few obstacles. With furniture, all methods are quite close together except for Neural Network which is always by far the slowest. All algorithms typically achieve a very high coverage between 96 and 99% in empty rooms, i.e. they all fulfill their major task. The best coverage is always achieved with the Contour Line method. With added furniture, the robot does not fit into every corner anymore and coverage drops to 92 to 96%. The examples in Fig. 4 indicate that uncovered areas are typically found in inaccessible corners or behind furniture.

Especially algorithms that aim to generate long and straight paths, like Boustrophedon, Convex SPP, Grid Local Energy, and Contour Line (esp. with few furniture), achieve a high appeal number. The Neural Network planner is least appealing due to its rather erratic trajectory with many intersections. Fig. 4 provides some qualitative impressions on the generated paths in three different environments. The results of all considered maps are available online [27].

We also executed the coverage trajectories obtained from the empty room maps on furnished maps with local pose adjustment for inaccessible path points if possible. This procedure achieved nearly the same performance values as reported on the unfurnished room maps in Tab. I for each method indicating that empty-room trajectories can be successfully applied to dynamically changing environments.

### B. Sensor Coverage Problem

In contrast to the last experiment, the sensor coverage problem is concerned with observing the room with a mid-range sensor with larger field of view and a center far away

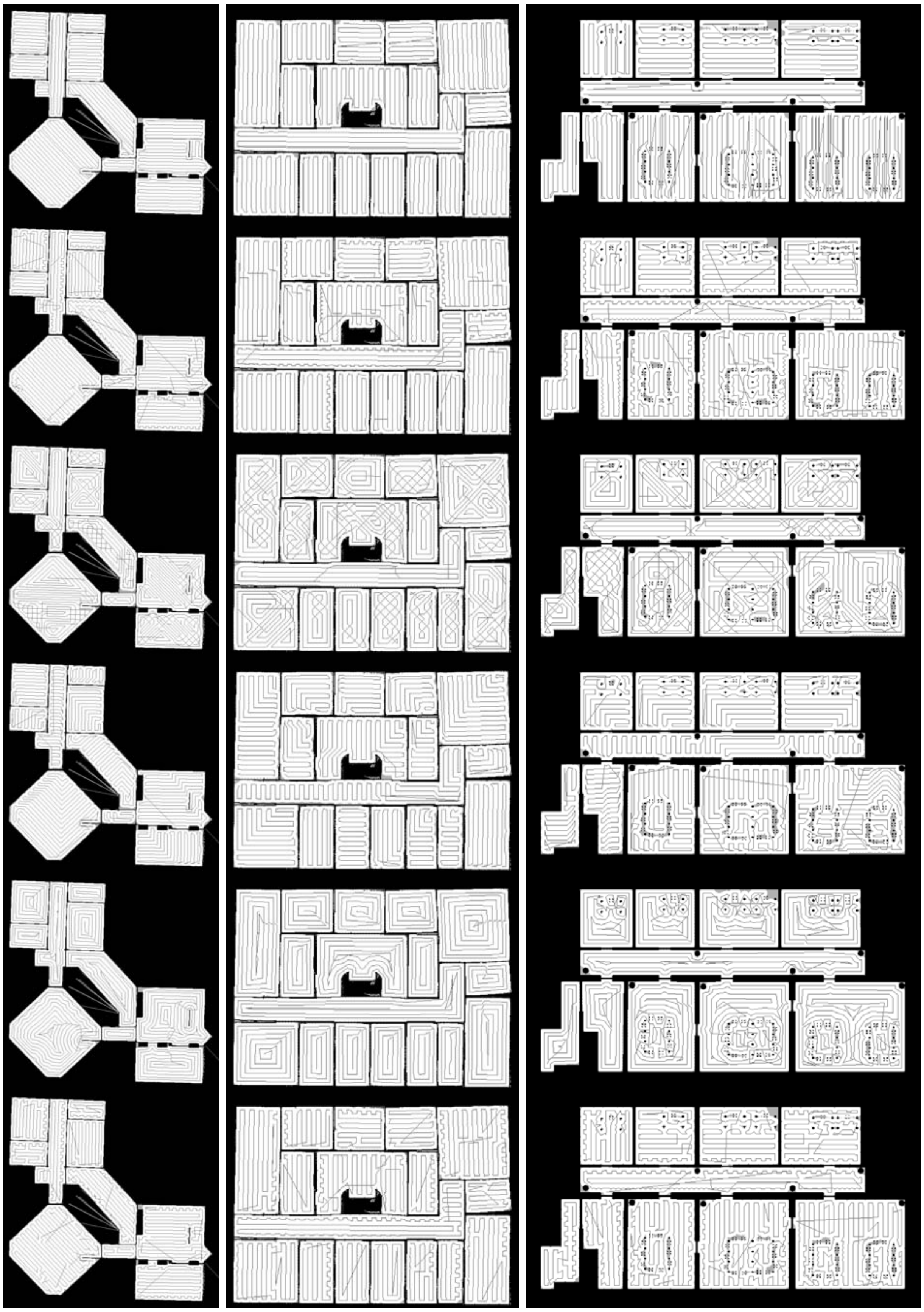

Fig. 4. Exemplary coverage paths: each row shows three exemplary maps processed with (a) the Boustrophedon, (b) Grid-based TSP, (c) Neural Network-based, (d) Grid-based Local Energy Minimization, (e) Contour Line, and (f) Convex Sensor Placement coverage path planning algorithms. The robot trajectories are displayed as dark gray lines, the covered area is white. Areas not covered are shaded with light gray.

| | Computation time [s] | Path length [m] | Rotations [rad] | Traveling time [s] | Coverage [%] | Appeal |
|---|---|---|---|---|---|---|
| without furniture | | | | | | |
| Boustrophedon | 1.8 (±0.8) | 101.6 (±166.2) | **53.5** (±64.5) | **440.7** (±651.1) | 98.4 (±3.9) | **1.219** (±0.130) |
| Grid TSP | 28.5 (±251.6) | 95.9 (±143.7) | 102.4 (±171.1) | 515.2 (±780.8) | 97.8 (±4.1) | 0.671 (±0.392) |
| Neural Network | 4.0 (±31.4) | 158.4 (±353.0) | 83.7 (±152.7) | 687.8 (±1414.8) | 96.4 (±5.7) | 0.108 (±1.752) |
| Grid Local Energy | **1.5** (±0.1) | **93.9** (±138.2) | 81.7 (±147.8) | 469.0 (±719.2) | 96.3 (±4.7) | 0.876 (±0.360) |
| Contour Line | 2.0 (±0.6) | 98.5 (±150.7) | 65.8 (±83.2) | 454.2 (±636.9) | **98.7** (±4.8) | 0.877 (±0.136) |
| Convex SPP | 26.0 (±215.3) | 94.7 (±141.4) | 99.3 (±156.6) | 505.2 (±745.3) | 98.4 (±4.0) | 1.078 (±0.308) |
| with furniture | | | | | | |
| Boustrophedon | 3.4 (±4.1) | 127.5 (±198.0) | **115.6** (±119.5) | 645.8 (±856.1) | 94.6 (±5.5) | **0.715** (±0.382) |
| Grid TSP | 41.6 (±259.0) | 95.2 (±145.0) | 168.5 (±197.5) | 639.3 (±849.1) | 94.4 (±5.5) | 0.416 (±0.308) |
| Neural Network | 4.0 (±32.0) | 215.8 (±380.9) | 171.6 (±204.1) | 1046.7 (±1606.7) | 92.9 (±6.9) | -1.049 (±2.624) |
| Grid Local Energy | **1.5** (±0.1) | 91.9 (±139.7) | 138.7 (±175.2) | **571.3** (±788.8) | 92.5 (±5.9) | 0.694 (±0.289) |
| Contour Line | 3.3 (±8.8) | 110.6 (±159.8) | 164.8 (±161.6) | 683.3 (±797.6) | **95.5** (±6.2) | 0.414 (±0.287) |
| Convex SPP | 39.5 (±230.9) | **91.1** (±140.9) | 157.2 (±182.5) | 603.9 (±807.7) | 94.7 (±5.5) | 0.657 (±0.254) |

TABLE I

AVERAGE QUANTITATIVE PERFORMANCE ON THE FOOTPRINT COVERAGE PROBLEM OVER ALL ROOMS AND STANDARD DEVIATION IN BRACKETS.

| | Computation time [s] | Path length [m] | Rotations [rad] | Traveling time [s] | Coverage [%] | Appeal |
|---|---|---|---|---|---|---|
| without furniture | | | | | | |
| Boustrophedon | 1.8 (±1.0) | 71.6 (±119.7) | 62.2 (±60.1) | 357.5 (±496.9) | 99.6 (±4.9) | 0.853 (±0.394) |
| Grid TSP | 5.6 (±37.2) | 61.4 (±93.8) | 88.8 (±128.7) | 374.1 (±549.4) | **99.8** (±1.7) | 0.796 (±0.332) |
| Neural Network | 2.0 (±7.5) | 94.2 (±276.1) | 68.6 (±108.7) | 444.8 (±1095.3) | 99.0 (±5.4) | 0.818 (±0.668) |
| Grid Local Energy | **1.5** (±0.3) | 54.7 (±82.2) | **59.7** (±94.5) | **296.5** (±445.1) | 98.9 (±4.7) | **1.082** (±0.257) |
| Contour Line | 2.1 (±0.8) | 59.8 (±103.3) | 68.8 (±68.3) | 330.8 (±460.0) | 98.2 (±4.5) | 0.849 (±0.189) |
| Convex SPP | 3.2 (±13.0) | **44.4** (±74.5) | 79.6 (±115.7) | 300.0 (±467.8) | 91.8 (±8.2) | 0.679 (±0.170) |
| with furniture | | | | | | |
| Boustrophedon | 3.3 (±3.8) | 93.7 (±154.0) | 104.8 (±117.2) | 512.7 (±712.9) | 97.4 (±5.5) | 0.523 (±0.388) |
| Grid TSP | 9.4 (±39.6) | 62.4 (±96.1) | 131.4 (±152.1) | 459.1 (±605.8) | 98.0 (±4.4) | 0.419 (±0.266) |
| Neural Network | 2.0 (±7.5) | 110.4 (±282.3) | 108.0 (±152.1) | 574.5 (±1179.3) | 96.1 (±8.5) | 0.322 (±1.289) |
| Grid Local Energy | **1.5** (±0.3) | 52.6 (±82.0) | 85.9 (±106.4) | 339.5 (±473.0) | 95.7 (±7.8) | **0.799** (±0.212) |
| Contour Line | 2.8 (±3.9) | 78.1 (±113.8) | 123.9 (±120.4) | 496.8 (±582.6) | **98.2** (±4.1) | 0.309 (±0.351) |
| Convex SPP | 3.7 (±13.3) | **40.5** (±73.4) | **78.9** (±119.6) | **285.8** (±471.2) | 88.5 (±9.8) | 0.620 (±0.179) |

TABLE II

AVERAGE QUANTITATIVE PERFORMANCE ON THE SENSOR COVERAGE PROBLEM OVER ALL ROOMS AND STANDARD DEVIATION IN BRACKETS.

from the robot footprint. For this experiment an Asus RGB-D sensor was assumed to be mounted at 65 cm height facing downwards to the ground yielding a trapezoidal field of view with small side length of 70 cm at 15 cm distance to the robot center and a large side length of 1.3 m at 1.15 m distance. Table II summarizes the performance measures for the 550 rooms with and without furniture.

Most of the aforementioned observations can be confirmed but due to the decreased dimensionality of problem space (via larger coverage radius) even the optimization algorithms are quite fast compared to the footprint coverage case. Nevertheless, the standard deviations reveal that only Boustrophedon, Grid Local Energy, and Contour Line reliably deliver results within their average computation times whereas the other approaches show high variance. Path lengths, rotations and traveling times generally decrease due to the larger coverage area compared to the previous experiment.

### C. Experiments with a Real Robot

We verified all 6 methods in our lab, which is the left map in Fig. 4 or *lab_ipa* in the dataset, using a modified Metralabs Scitos G5 robot which was equipped with a wet cleaning device. In all cases, the movement pattern of the robot matched the computed target trajectories well so that complete coverage with the wet cleaner could be established. According to the consulting of a large professional cleaning service provider the Boustrophedon planner provides the best potential for usage with professional wet cleaning machines. The robot and an exemplary trajectory of the Boustrophedon planner are shown in Fig. 5.

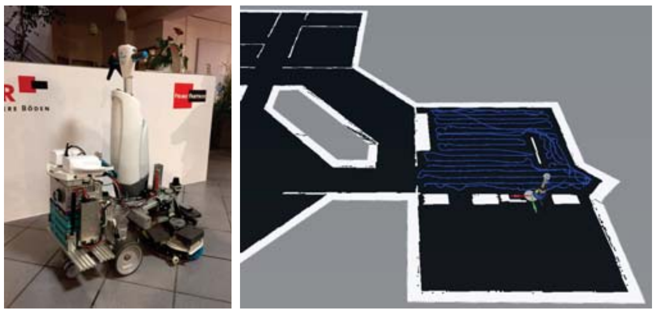

Fig. 5. The prototypical robot (left) used for verifying the computed trajectories in our lab, map *lab_ipa* (right).

## V. CONCLUSIONS

The paper has introduced and compared several algorithms suited for indoor coverage path planning. It became obvious that due to varying requirements different algorithms can be suited best for this problem, either because of their computation speed, their high coverage, or the beauty of their generated paths. We conclude with some general remarks on some of the algorithms. First, although the Boustrophedon planner generates the most appealing paths with good performance for nearly empty rooms its quality degrades with increasing clutter. The reason is the cell decomposition which becomes too detailed in such cases. A better cell decomposition with suitable split and merge criteria is on our research agenda. Furthermore, complementing the Boustrophedon planner with the the Grid Local Energy planner, which is very strong in cluttered areas, could be a successful path to go. Second, the Grid-based TSP determined its traveling costs as A$^*$ map distances, however, does not yet consider the additional costs of turning the robot. In future work it could be interesting to incorporate rotation costs directly in the TSP path planning, enabling the direct optimization of traveling times.

Third, using the parameters of the original publication for the Neural Network [19] we could not observe similar movement patterns as shown in their paper as the robot does not intend to stay close to already visited areas. A drawback of the Neural Network is the need for a dense grid since it cannot jump over larger gaps as the Grid-based Local Energy Minimization can do, for example. An extension with a dynamic neighborhood adaptation could mitigate this problem. An advantage of the Neural Network and Local Energy Minimization approaches is the online applicability to incomplete or dynamic maps.

To wrap up, if a quick heuristic is needed because of processing constraints, the Boustrophedon, Grid-based Local Energy Minimization and Contour Line-based Coverage Path Planning methods are good choices with individual strengths. If time allows, the Convex Sensor Placement is an interesting close-to-optimal alternative especially for the sensor-based observation use case with very short robot traveling times if potentially erratic paths are acceptable, which is typically the case in inspection tasks.

## REFERENCES

[1] F. Yasutomi, M. Yamada, and K. Tsukamoto, "Cleaning robot control," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Apr. 1988, pp. 1839–1841.

[2] C. Hofner and G. Schmidt, "Path planning and guidance techniques for an autonomous mobile cleaning robot," *Robotics and Autonomous Systems*, vol. 14, no. 23, pp. 199 – 212, 1995.

[3] M. Bosse, N. Nourani-Vatani, and J. Roberts, "Coverage Algorithms for an Under-actuated Car-Like Vehicle in an Uncertain Environment," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Apr. 2007, pp. 698–703.

[4] M. Ollis and A. Stentz, "Vision-based perception for an automated harvester," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, Sept. 1997, pp. 1838–1844.

[5] R. Bormann, J. Hampp, and M. Hägele, "New Brooms Sweep Clean - An Autonomous Robotic Cleaning Assistant for Professional Office Cleaning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2015.

[6] 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. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092188901300167X

[7] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele, "Room Segmentation: Survey, Implementation, and Analysis," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[8] H. Choset, "Coverage for robotics - A survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1, pp. 113–126, 2001. [Online]. Available: http://www.springerlink.com/index/q346pp34036143pg.pdf

[9] M. A. Arain, M. Cirillo, V. H. Bennetts, E. Schaffernicht, M. Trincavelli, and A. J. Lilienthal, "Efficient measurement planning for remote gas sensing with mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3428–3434. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7139673/

[10] J.-C. Latombe, "Exact Cell Decomposition," in *Robot Motion Planning*, ser. The Springer International Series in Engineering and Computer Science. Springer US, 1991, no. 124, pp. 200–247, dOI: 10.1007/978-1-4615-4022-9_5. [Online]. Available: http://link.springer.com/chapter/10.1007/978-1-4615-4022-9_5

[11] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Proceedings of the International Conference on Field and Service Robotics*. Springer, 1998, pp. 203–209. [Online]. Available: http://link.springer.com/chapter/10.1007/978-1-4471-1273-0_32

[12] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *Journal of Field Robotics*, vol. 26, no. 8, pp. 651–668, Aug. 2009. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/rob.20300/abstract

[13] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse Decompositions for Coverage Tasks," *The International Journal of Robotics Research*, vol. 21, no. 4, pp. 331–344, Apr. 2002. [Online]. Available: http://journals.sagepub.com/doi/abs/10.1177/027836402320556359

[14] S. C. Wong and B. A. MacDonald, "A topological coverage algorithm for mobile robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, Oct. 2003, pp. 1685–1690.

[15] H. Choset, E. Acar, A. A. Rizzi, and J. Luntz, "Exact cellular decompositions in terms of critical points of morse functions," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3. IEEE, 2000, pp. 2270–2277. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/846365/

[16] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 2001, pp. 27–32.

[17] A. Zelinsky, R. A. Jarvis, J. C. Byrne, and S. Yuta, "Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot," in *In Proceedings of International Conference on Advanced Robotics*, 1993, pp. 533–538.

[18] Y. Gabriely and E. Rimon, "Spiral-STC: an on-line coverage algorithm of grid environments by a mobile robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 2002, pp. 954–960.

[19] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 718–724, Feb. 2004.

[20] T. Shermer, "Recent results in art galleries," *Proceedings of the IEEE*, vol. 80, no. 9, pp. 1384–1399, Sep 1992.

[21] "Room segmentation dataset and software," http://wiki.ros.org/ipa_room_segmentation.

[22] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[23] S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica*, vol. 2, no. 1-4, pp. 153–174, 1987.

[24] H. Choset, "Incremental construction of the generalized voronoi diagram, the generalized voronoi graph, and the hierarchical generalized voronoi graph," in *Proceedings of the First CGC Workshop on Computational Geometry*, 1997.

[25] D. Applegate, R. Bixby, V. Chvatal, and W. Cook, "Concorde TSP solver," http://www.math.uwaterloo.ca/tsp/concorde.html, 2006.

[26] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, Feb. 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370297000787

[27] "Documentation on ipa_room_exploration software," http://wiki.ros.org/ipa_room_exploration.