2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Systematic Floor Coverage of Unknown Environments Using Rectangular Regions and Localization Certainty

Dhiraj Goel James P. Case Daniele Tamino Jens-Steffen Gutmann
Mario E. Munich Mike Dooley Paolo Pirjanian

***Abstract*— We address the problem of systematically covering all accessible floor space in an unknown environment by a mobile robot. Our approach uses rectangular regions that are swept across the environment. In the first stage, the robot covers each region using the classic boustrophedon pattern and planning paths to uncovered areas within the region while keeping track of its position uncertainty. The region is then moved sideways to cover the next part of the environment until all accessible space has been visited. In the second stage, the robot revisits the perimeter around the obstacles. We compare our method in terms of total trajectory length to 5 off-line methods including the distance transformation by Zelinsky *et al.* [1] in a standard test environment as well as in multi-room homes. The presented method has been employed in our Mint cleaning robot [2] for autonomously sweeping and mopping the floors.**

I. Introduction

In this paper we address one of the classic problems in mobile robot navigation: the coverage of all accessible floor space in an unknown environment by a mobile robot. Floor coverage is important, for example, when comparing the performance of autonomous robotic floor cleaners [3], [4], [5].

The main question that arises is how to navigate to achieve complete coverage? For example, the robot can follow a random walk pattern like the motion of a Roomba vacuum cleaner. A different approach is to follow the perimeter of an area first and fill in the inner area afterwards. This has been proposed by Ulrich *et al.* [6] and is employed in Neato Robotics' robotic vacuum cleaner.

In general, complete coverage is related to the traveling salesman problem and finding an optimal solution is NP complete [7], even if the environment is known in advance. Since the environment is unknown, though, the problem is also related to robot mapping and exploration [8], [9], [10].

An overview of earlier methods for coverage can be found in the article by Choset [11]. One of the first algorithms is the distance transformation by Zelinsky *et al.* [1]. A wave front is initiated from a goal to a start pose. The robot then follows the steepest ascent of the wave which brings it furthest away and slowly towards the goal. We use this approach as one of the off-line methods in our experiments.

Other classic approaches are Choset's work on the *boustrophedon cellular decomposition* [12], [13]. It divides an environment into cells using an exact decomposition. Each cell is covered using back and forth motions, i.e. the *boustrophedon path*. The method then moves to the next cell or backtracks until all cells are covered. Choset also suggested to follow the perimeter of obstacles a second time after completing the main coverage in order to fill in gaps between obstacles and the boustrophedon path.

Choset's approach has been extended in various ways. Huang pointed out that turning takes more time than moving straight and formulated the exact cell decomposition with different sweep directions that allow for coverage paths with less turns [14]. Acar and Choset refined the boustrophedon decomposition using Morse functions and provided methods more robust to sensor noise [15], [16], which were also used and further enhanced by Garcia and Gonzalez [17]. Our method makes use of some of these concepts when the robot senses obstacles and needs to decide whether to turn around or follow along the obstacle.

Wong and MacDonald proposed a topological coverage algorithm using natural landmarks extracted from the environment and an exact cell decomposition approach [18].

Kang *et al.* proposed a cell decomposition on a known occupancy map of the environment and allow rotations of the map before performing the regular line sweeping operation for finding critical points [19]. They use a set of pre-defined template paths for covering the cells. In a later work they divide the known occupancy map into regions by detecting small openings using the Voronoi graph and covering each region separately before moving to the next one [20].

Recently Viet *et al.* combined the boustrophedon cell decomposition with an A* path planner for finding shortest paths to the next target location [21].

There is also work on finding optimal solutions to the coverage task. Gabriely and Rimon assume an occupancy grid representation of the environment with each cell divided into four equally sized sub cells. They compute a spanning tree over the grid and circumnavigate the tree which visits each sub cell exactly once [22]. They later also formulated a spiral motion path for covering an environment [23].

Mannadiar and Rekleitis compute a Reeb graph from the boustrophedon cellular decomposition of a known environment and find the optimal path that guarantees complete coverage as an Euler tour by splitting some of the cells into two components [24]. Xu *et al.* extended this method for non-holonomic robots by minimizing the distance traveled over previously covered regions [25].

Most of the previous work assume the robot is localized at all times. A notable exception is the method by Das *et al.* [26] who formulated coverage with position uncertainty. We make explicit use of the position uncertainty of the robot and control the coverage method accordingly.

Our approach uses an occupancy grid to represent the environment which is learned as the robot covers the space. A localization system provides pose estimates of the robot and is typically a SLAM method that can employ active beacons [27], [28] or a vision system [29]. We assume that the robot

The authors are with iRobot Corp., 1055 E. Colorado Blvd., Suite 340, Pasadena, CA 91106, USA.

978-1-4673-6357-0/13/$31.00 ©2013 IEEE

has good relative pose estimation, e.g. from odometry and a gyro, and a global uncertainty about its pose with respect to a fixed reference frame. The relative pose estimates allow the robot to locally cover an area without leaving gaps or covering the same area multiple times, while the global uncertainty tells the robot how far to continue when moving into areas where localization becomes less certain.

We place a rectangular region over the occupancy grid and let the robot cover as much accessible space as possible within the region. The region is successively swept over the occupancy grid until the complete environment has been covered or further exploration is prevented by uncertainty in robot position.

By limiting the robot's coverage within a region we avoid moving back and forth between distant locations. To a human observer this behavior looks intelligent and makes the navigation of the robot more predictable. We believe this is an important aspect for a consumer product that has not received much attention in previous coverage approaches.

After completing all regions, the robot revisits the boundaries of obstacles and covers their perimeter a second time. This is important since the areas in proximity of the obstacles are much harder to navigate as compared to open space, and therefore gaps in coverage are more likely to occur. Furthermore, for a cleaning robot, revisiting the obstacle perimeter is advantageous as dust and dirt often accumulate there.

We evaluate our coverage method on the Mint cleaning robot [2] in a standard test environment and in several homes with floor sizes up to $125\,\mathrm{m}^2$, and compare it to 5 off-line methods that assume a given map, have exact localization and achieve full coverage. The results show that our method performs competitively in terms of coverage and trajectory length.

The rest of this paper is organized as follows. In the next section we describe our coverage algorithm using rectangular regions and perimeter following. In Section III we perform extensive experiments for evaluating our approach and compare it to other methods. We conclude in Section IV.

II. Coverage Using Rectangular Regions

Our coverage approach uses two stages. In the first stage open area is covered by placing rectangular regions over the environment and mapping the space. In the second stage the perimeter of obstacles is visited one more time. In the following sections we describe the occupancy grid representation, the region parametrization, region coverage, region placement and and the perimeter following strategy. We also present refinements to the overall strategy for a practical system.

A. Occupancy Grid

We place a regular occupancy grid over the environment. Each cell in the grid can take one of the values *unknown*, *covered*, or *obstacle*. Initially all cells are set to unknown since the environment is not known *a priori*.

For *covered* cells we also store a number representing the certainty in the robot position estimate obtained from localization. This can be obtained from the position covariance. Note that this value depends on the trajectory of the robot and we keep the maximum certainty number at each covered cell. In general though, cells in areas with poor localization tend to have low certainty number.

For *obstacle* cells we also distinguish between different types of obstacles depending on how they are sensed. For example, impassable obstacles (detected by bumper or wheel motor stall), cliffs (detected by drop sensors) and hazards (detected by robot being physically wedged or stuck).

The initial position of the robot is at the origin of the occupancy grid. The cells traversed by the robot (center of the wheelbase) are set to *covered* with associated instantaneous position certainty. When detecting an obstacle the cell closest to the obstacle is marked accordingly. When the robot gets closer to one of the dimensions of the occupancy grid, the map is re-centered to fully utilize the map. This operation is also done opportunistically.

B. Rectangular Region

We place a rectangular region over the occupancy grid. Fig. 1(a) shows our setup and names several parameters.

The rectangular region is parameterized by its location, width w_b and length l_b. The robot moves back and forth within this region using the classic boustrophedon pattern. We call each longitudinal segment of the boustrophedon path a *rank*. A rank has a start pose, a rank length l_r and a progress direction (either left or right). The robot moves along a rank and after reaching the endpoint, it turns onto the next rank. The distance between neighboring ranks is defined as rank width w_r.

The rank width w_r should be less than or equal to the width of the robot's cleaning head w_t in order to leave no area uncovered between the ranks. Preferably, w_r should be smaller than w_t so that the ranks overlap, allowing for small errors in control and localization [26]. For example, $w_r = 0.7 \times w_t$.

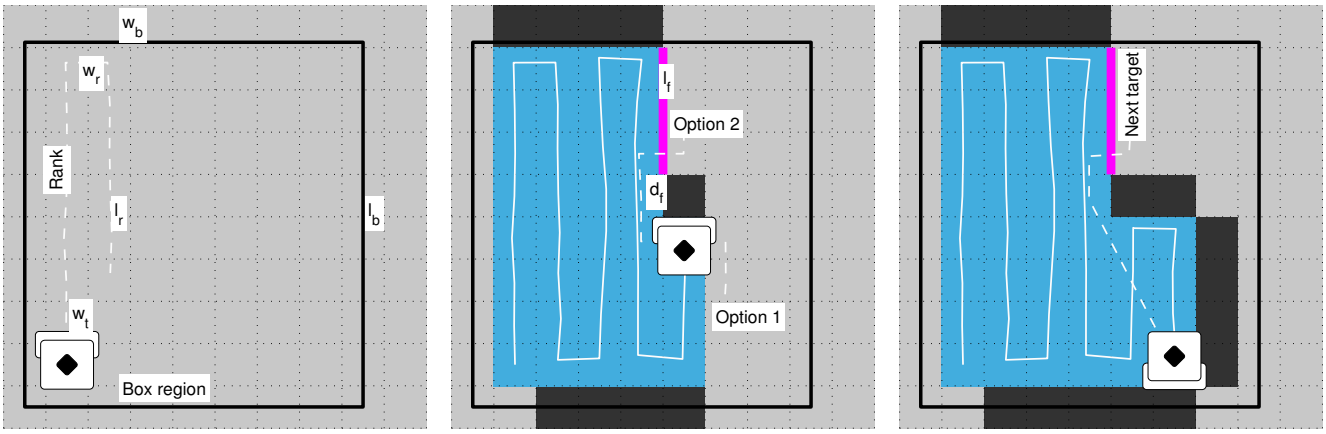
The rank length l_r can vary but is limited to a maximum rank length $\hat{l}_r$. This ensures that the robot does not move arbitrarily far away from where it started the boustrophedon pattern. It also limits the amount of position error the robot accumulates from its relative motion estimates. Ideally both rank width w_r and maximum rank length $\hat{l}_r$ should be chosen such that the chance of leaving uncovered area between ranks is minimal, the overlap of ranks is small, and the rank length is large.

When placing a rectangular region, its width w_b is fixed but the length l_b can vary up to a maximum $\hat{l}_b$ by extending the region at the top or the bottom. This allows the region to adapt to the physical dimensions of the environment. Limiting the maximum length ensures that the robot only navigates in a local area defined by the region even in a large environment. We allow $\hat{l}_b$ to be larger than the maximum rank length $\hat{l}_r$.

The ideal cell size of the occupancy grid is the rank width w_r. This corresponds to each new rank filling in exactly one column in the occupancy map. Alternatively, the cell size could be chosen such that each rank fills in an integral number of columns.

C. Basic Region Coverage

The initial rectangular region is placed such that the robot is in the bottom left corner as shown in Fig. 1(a). The robot moves forward on the first rank and places new ranks to



(a) Rectangular region over occupancy grid (b) Robot encounters an obstacle (c) Robot reaches end of boustrophedon path

Fig. 1. Floor coverage using rectangular regions. Blue areas indicate covered space. Robot trajectory is drawn in white with intended future path as dashed line.

the right. As the robot moves along a rank, the region is expanded until it reaches the maximum length $\hat{l}_b$.

When following a rank, if the robot reaches the maximum rank length $\hat{l}_r$ or the end of the region, it stops and turns onto the next rank. On the other hand, if it encounters an obstacle, it backs up and evaluates two options in order to continue. Fig. 1(b) illustrates the scenario.

One option is to back up from the obstacle and just turn onto the next rank. Another option is to follow the obstacle over covered cells, towards the previous rank, and attempt to merge back onto the current rank behind the obstacle. The second option is only possible if there are unknown cells, next to the covered cells, behind the obstacle. We call the boundary between unknown and covered cells an *open frontier* (magenta line in Fig. 1(b)). Only covered cells with a position certainty larger than a threshold qualify for becoming an open frontier. This discards areas with low position certainty. The criteria for deciding between the two options are the distance d_f from the current robot pose to the frontier, and the ratio of frontier length l_f and current rank length l_r. If the frontier is close and if l_f is larger than l_r then option 2 is preferred.

D. Finding Next Target Rank

At some point the robot is not able to continue its boustrophedon path because it is at the end of the region, or it's position certainty is too low, or there are obstacle cells blocking the next rank (example in Fig. 1(c)).

At this point, the robot searches for open frontiers in the rectangular region and chooses the next target rank and start location. In many cases (e.g. the one in Fig. 1(c)) both ends of a frontier are considered as candidates. In order to decide which one to go to, a path is planned to each candidate location. For path planning we use a wave front method that computes the distance of each cell to the robot position in grid coordinates similar to the brush-fire algorithm [30]. A wave is started from the cell of the robot position. Covered cells that are next to a wave cell are added to the wave such that the grid is searched breadth-first until all candidate cells are found or all wave cells have been examined. The robot then chooses the candidate with the minimum path distance. When following a path, we find collision-free shortcuts along the path cells such that the robot travels on straight lines

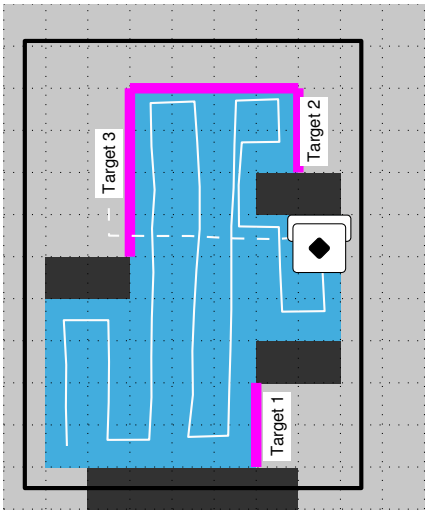


Fig. 2. Robot reaches end of rectangular region and has multiple candidate locations as the next target. Note that the middle ranks reached the maximum rank length (which caused the robot to turn around) and thus, the top frontier spans multiple grid columns.

at arbitrary orientations. The robot never plans a path over unknown cells as there may be obstacles causing additional maneuvers or re-planning.

In general, there can be multiple open frontiers in the rectangular region that the robot can choose from. Additionally, frontiers can span more than one column in the occupancy map. Fig. 2 shows a more complex scenario with two frontiers.

We split frontiers into multiple components by fitting straight line segments aligned to the coordinate axes of the region, and discarding the ones with very small projected length along the main coverage direction. In the scenario of Fig. 2 the top frontier is split into three components but only the left and right components are kept as targets. For each target, both the two endpoints are considered as potential start locations for a new rank. The robot chooses the one with the shortest path distance. For example, in Fig. 2, the robot selects the lower endpoint of Target 3 because it's path distance (5 cells) is smaller than the lower endpoint of target 2 (6 cells) and the upper endpoint of Target 1 (7 cells). Note that at the end of each rank at Target 3, the robot

```
start:
  move forward
loop
  if obstacle in front then
    if frontier behind and small d_f and l_f > l_r then
      follow around obstacle on previous rank
      next loop
    end
  else if in rectangular region and l_r < l̂_r then
    move forward
    next loop
  end
  if position certain and in rectangular region and no
  obstacle on next rank then
    turn onto next rank
    next loop
  end
  find open frontiers in rectangular region
  if found then
    follow path to closest target rank
    next loop
  end
  stop    // done with rectangular region
```

Algorithm 1: Covering a rectangular region

will progress to the left to start the new rank. This progress direction is determined by the side on which unknown cells are located with respect to the target rank.

The process of basic region coverage using boustrophedon paths and finding next target ranks is iterated until no further targets are found. Algorithm 1 summarizes this approach.

E. Region Placement

After a region is completed, it is moved by $w_b - w_r$ to one side for covering the next area of the environment while keeping the same dimensions l_b and w_b. The overlap of at least one rank width between consecutive regions ensures that no frontier is missed while switching regions.

In the beginning the rectangular region is moved towards the right until a maximum distance from the initial start position is reached or no more open frontiers are found (robot is at the boundary of the environment). The rectangular region is then placed to the left of the first region and successively further left for covering the other side of the environment.

After rectangular regions have been placed on both sides, the whole occupancy grid is examined for further open frontiers and new regions are placed accordingly. In this case we also allow regions to be oriented horizontally, i.e. the robot can move along a horizontal boustrophedon path.

F. Perimeter Follow

Once no more open frontiers are found, the robot enters the second stage where it visits the perimeter of obstacles a second time. This stage serves two purposes as far as coverage is concerned. First, to cover areas next to the obstacles that might have been missed by the boustrophedon path. Second, to tightly follow narrow passages or small openings that can lead to previously unexplored areas that were not uncovered in the open area phase by the boustrophedon path due to it's discrete pattern.

```
start:
  place rectangular regions to the right
  place rectangular regions to the left
  find open frontiers in full map
repeat
  while open frontier not completed do
    place rectangular region at closest target
  end
  find perimeter frontiers
  while perimeter frontier not completed do
    follow perimeter of closest target
  end
  find open frontiers in full map
until no open frontier left
follow path to start pose
```

Algorithm 2: Overall coverage method

Covered cells in the occupancy grid that are next to obstacle cells and whose localization certainty is above a certain threshold are marked as a *perimeter frontier*. We exclude cells that have low localization certainty since the chance of successfully navigating there is low.

After finding all perimeter frontiers we use a similar strategy as for open frontiers. The endpoints of all frontiers are evaluated according to their path distance from the current robot position and the closest one is chosen as the next target. For following the perimeter, the robot may use dedicated wall sensors and a wall-follow behavior that keeps the robot close to walls and furniture.

During wall following the robot may explore uncovered areas. Therefore, after completing all perimeter frontiers, we return to the first stage where open frontiers in the occupancy grid are searched for. If open frontiers are found, additional rectangular regions are placed and covered by the robot.

The process of open area coverage and perimeter follow is iterated until no further open or perimeter frontiers are left to cover. At this time the floor coverage is completed and the robot returns to its start location (see Algorithm 2).

G. Refinements for Practical Application

The coverage algorithm outlined above systematically covers an environment. For practical reasons, we made several refinements that make the robot more predictable to an observer and reduce the amount of time spent in driving back and forth between distant places.

1) Minimum frontier length: Frontiers that do not meet a minimum length are discarded. This speeds up progress in coverage by ignoring small details at the cost of possibly not fully covering an area. Often, small frontiers are covered later when following the perimeter of obstacles.

2) Defer coverage of distant frontiers: If the path distance to an open frontier in the rectangular region becomes too large, we defer its coverage and continue with the next region. This happens for example when an obstacle crosses the entire rectangular region and disconnects areas. Often, it is more efficient to first cover the connected areas. The deferred, uncovered areas are visited later when examining the full occupancy map.

3) Active localization: As the robot covers the environment it keeps track of its position uncertainty. In the event that the robot becomes too uncertain, it pauses the coverage planning and returns back to a location where its sensors obtained good information for localization. After improving it's position certainty, the robot then continues with the floor coverage. This strategy allows the robot to explore deeper into areas of the environment that have poor localization.

4) Overlapping rectangular regions: When placing the first rectangular region to the left side, we overlap it with the very first region such that several columns in the occupancy grid are covered a second time. The intend is to allow for some localization error when the robot comes back for covering the left part of the environment, and to let the robot traverse areas where it can obtain good information for localization.

5) Credit for wall follow: Whenever the robot is following a wall over a longer distance during open area coverage in the first stage, we give credit and don't consider that obstacle segment for perimeter follow in the second stage. Note that quite often the robot in stage 2 still has to travel close to the wall again when visiting other perimeter frontiers connected to the wall.

6) Interleave open area coverage and perimeter follow: We can perform perimeter follow before completing all open areas by also bounding the search on perimeter frontiers to a rectangular region. We found that performing perimeter follow after completing each side of the environment to be quite effective since the robot is still in the area it has just covered.

III. Results

We evaluate our approach to floor coverage using rectangular regions on the Mint robot. Mint is an autonomous and systematic floor cleaner that sweeps and mops floors [2]. It uses the Northstar system [31] for measuring directions to IR spots projected onto the ceiling by Northstar cubes. By learning the spatial distribution of the IR signals, Mint can localize itself around a beacon [27]. The localization area can be expanded by using multiple projector cubes [28].

We ran Mint in two different categories of environments. One is a standard test room that is under discussion at the International Electrotechnical Commission (IEC) for evaluating the navigation performance of robotic vacuum cleaners. The other one is a set of multi-room home environments.

For evaluation two metrics are employed that were previously proposed by others [4]: effectiveness as percentage of area covered, and efficiency as the total distance traveled.

We also compare our coverage strategy to five other methods that use an *a priori* map of the environment, assume perfect sensing and localization, and simulate the coverage path.

The first four methods are greedy strategies. Algorithm 3 shows the strategy *forward* which borrows its name from the fact that the method always tries to move forward first. Turn left and turn right motions make the robot rotate 90 degrees and move forward to the next cell. For finding unknown cells we use the same wave front method as described in Section II and take obstacle-free shortcuts in the found path. Algorithm 4 shows a variant which we term *U-turn* as it tries to turn twice when turning in one direction. This resembles

```
if cell in front is unknown then
    move forward
else if cell on the left is unknown then
    turn left
else if cell on the right is unknown then
    turn right
else
    find closest unknown cell
    if found then
        follow path to found cell
    else
        stop
    end
end
```

Algorithm 3: Greedy strategy *forward*

```
if last-turn was left and cell on the left is unknown then
    turn left and clear last-turn
else if last-turn ws right and cell right is unknown then
    turn right and clear last-turn
else
    use Algorithm 3 and remember turn as last-turn
end
```

Algorithm 4: Greedy strategy *U-turn*

```
if orientation is horizontal then
    use Algorithm 3
else if cell on the left is unknown then
    turn left
else if cell on the right is unknown then
    turn right
else
    use Algorithm 3
end
```

Algorithm 5: Greedy strategy *horizontal*

the pattern of the boustrophedon path. Strategy *horizontal* in Algorithm 5 makes the robot follow a horizontal boustrophedon path while strategy *vertical* (analogous to Algorithm 5) moves the robot vertically. It is worth noting that all four methods make use of the known map only for sensing cells and can, principally, be also run without an *a priori* map and on-board the robot.

As a fifth method we implemented the distance transformation from Zelinsky *et al.* [1]. A wave front initiated at the start location computes the path distance to all cells in the grid map. The robot then follows the steepest ascent and covers the cells most distant from the start location first.

All 5 methods first cover the entire environment and then follow the perimeter of obstacles a second time like in our coverage method using rectangular regions. This is achieved by setting all cells next to obstacles back to unknown and running the coverage method another time. At the end each method also navigates the robot back to its start position.

Note that the comparison of the five off-line methods to our coverage method using rectangular regions is not an apples-to-apples comparison. Our on-line method runs on the robot and makes use of the refinements in Section II-

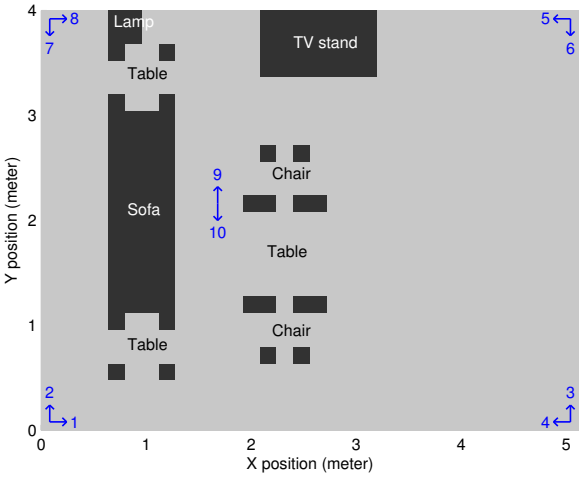


Fig. 3. IEC room with 10 different start positions and orientations.

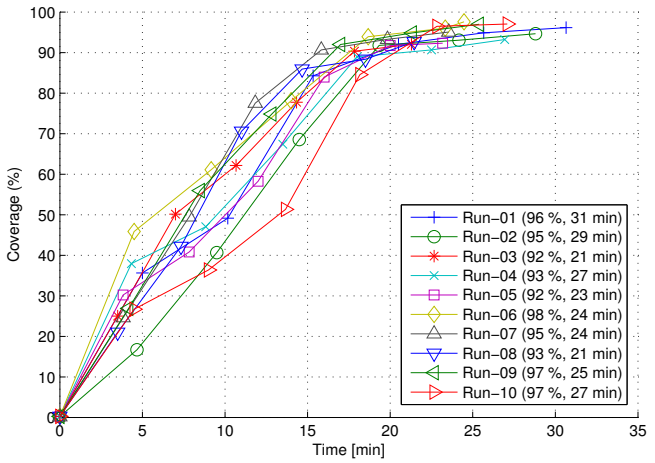


Fig. 4. Coverage over time when running Mint from all 10 start poses. Final percentage and cleaning time are in parentheses.

G while the off-line ones follow the strict rules described above. Some of the refinements, i.e. minimum frontier length, credit for wall-follow and interleaving perimeter follow with open area coverage, create an advantage for the on-line method while others, i.e. active localization and overlapping regions, as well as the fact that the robot does not have access to an *a priori* map and has to deal with sensor and localization uncertainties, do not. We believe that, in general, the comparison to the off-line methods still provides a good indication of how well our coverage method performs.

For computing the distance traveled by the robot we use the following cost functions for the five off-line methods. For moving into the next cell the robot has to traverse a cell width. For Mint this is $w_r = 18\,\text{cm}$. When turning, the robot with wheelbase $b = 17\,\text{cm}$ can turn in place (cost $\frac{1}{4}b\pi$) or pivot on one wheel (cost $\frac{1}{8}b\pi$). We use the average of the two, thus $\frac{3}{16}b\pi$. For the first four off-line methods we also consider an obstacle cost. When trying to move forward and the cell in front is an obstacle, the robot has to move forward for sensing the obstacle and backup afterwards for allowing to turn. We approximate this by a cost of 10 cm. The distance traveled by the actual robot is computed by accumulating the encoder ticks of both wheels, taking the average, and multiplying it by a fixed scale factor which has been determined off-line.

A. IEC Room

Fig. 3 shows an occupancy map of an earlier version of the IEC room for measuring the coverage of a cleaning robot. Obstacles are drawn from the perspective of the robot, e.g. for tables and chairs only the legs are visible. There are 10 start poses. We placed a Northstar beacon in the room center such that localization is certain over the whole environment. We also installed a ground truth system using motion capture cameras that tracks the robot as it moves.

Fig. 4 shows the coverage over time when running Mint from all 10 start positions. The cleaning time ranges from 21 min to 31 min and the total area covered is between 92 % and 98 % of the accessible space. The robot trajectory when started from position 10 is displayed in Fig. 6(a) as observed by the ground truth system.

When running the five off-line methods on the occupancy map of the IEC room from position 10, the trajectories in Fig. 6(b) to Fig. 6(f) are obtained. Note that for illustration we added small Gaussian noise onto the trajectory positions in order to improve visibility of overlapping path segments.

The forward strategy makes the robot move in a spiral pattern. The U-turn strategy places the boustrophedon pattern horizontally and vertically while the other two greedy methods cover the space mostly horizontally and vertically. The distance transformation approach sends the robot out to the farthest place and then tracks back in waves towards the start. Note that some of the strategies, e.g. U-turn and the distance transformation result in trajectories that may not look predictable to an observer.

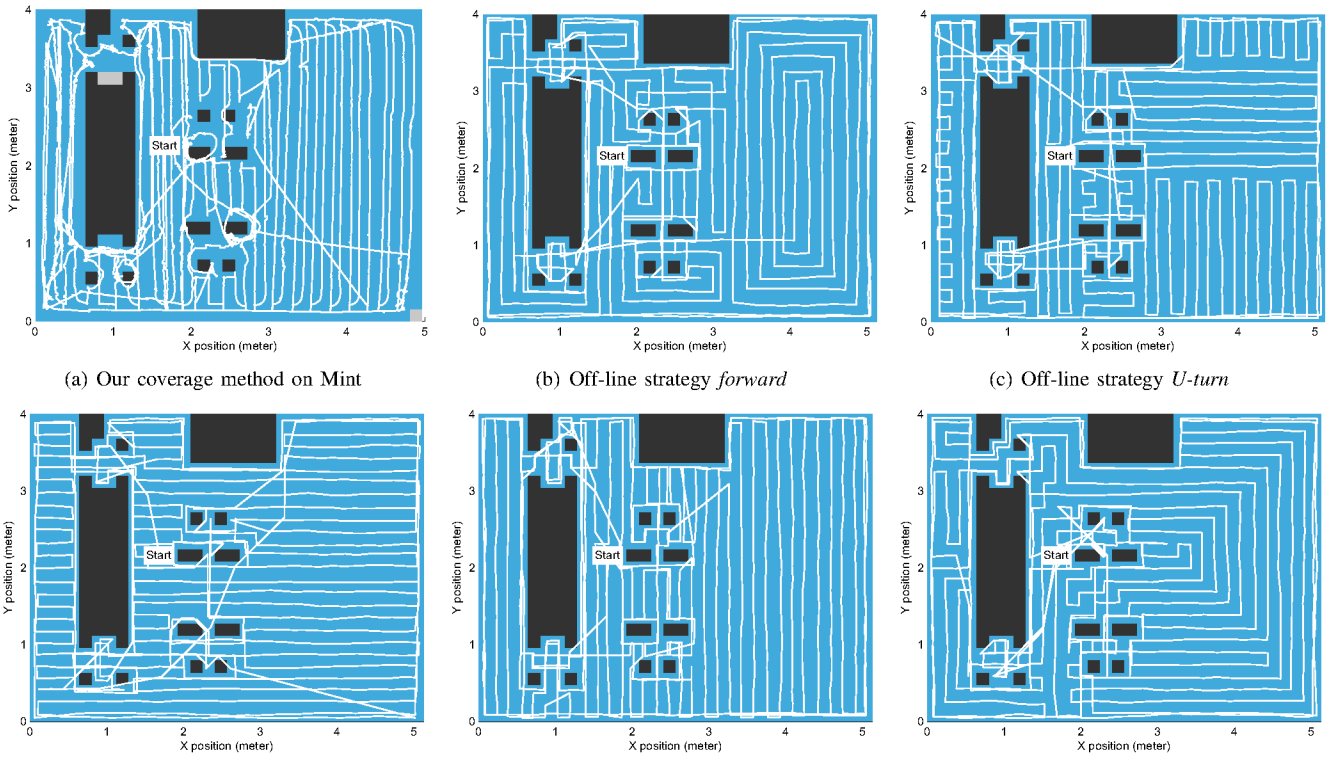
Fig. 5 compares the distance traveled of Mint and the five off-line methods over all 10 start positions. The forward strategy has the least variable travel distance with a mean of about 195 meters. The distance transformation varies more but has a similar average. Note that the distance transformation does not include the obstacle cost defined earlier since the method computes a complete path on the *a priori* occupancy map. We use the performance of the distance transformation as a baseline and show the travel distances of the other methods as factors in parentheses. Our coverage method using rectangular regions compares relatively well and uses only about 9% longer travel distances. Note that the outlier at start position 8 is due to the robot only covering 93 % of the IEC room.

B. Multi-Bedroom Home Environments

We also ran Mint in several multi-room home environments with up to four Northstar cubes that extend localization to the larger space of up to $125\,\text{m}^2$. Fig. 8 shows the final occupancy map obtained by Mint in one of the home environments.

A total of 12 runs were carried out in this and other homes. Using the final map obtained by Mint, we ran our five off-line methods for comparison. We performed some minor editing on the maps to ensure that all accessible areas are connected so that all cells are covered by the off-line methods. For perimeter follow, the off-line methods only considered the obstacles discovered in the map during open area coverage.

Fig. 7 shows the performance of all methods. Again we use the distance transformation as a baseline and compute the average distance of all coverage methods as a factor. The



(a) Our coverage method on Mint (b) Off-line strategy *forward* (c) Off-line strategy *U-turn*

(d) Off-line strategy *horizontal* (e) Off-line strategy *vertical* (f) Distance transform [1] on *a-priori* map

Fig. 6. Robot trajectories of all coverage methods in the IEC room when starting the robot from start pose 10.

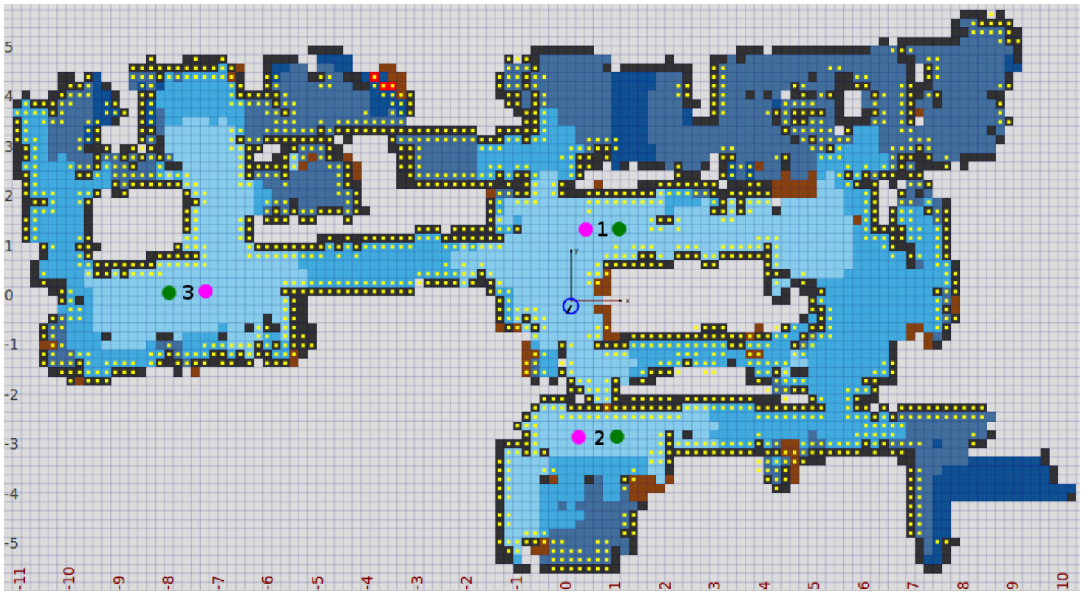


Fig. 8. Occupancy grid generated in a three bedroom house with three Northstar cubes. Mint cleaned about $125\mathrm{m}^2$ in 3 hours. Green and magenta discs show the estimated position of Northstar spots. Shades of blue indicate covered cells where the lighter shades correspond to higher position certainty. Obstacles are drawn in black (bumper or wheel stall), brown (cliff) and red (hazard). Obstacles that are perimeter-followed are marked by yellow dots. Cell size is 18 cm. Units are in meters.

greedy methods are now slightly worse than the distance transformation. Despite the more challenging environment our approach still performs quite well as the increase in average travel distance is only 9 % compared to the baseline.

IV. Conclusion

We presented a method for coverage using rectangular regions that are swept across the environment. Each region is covered systematically using the boustrophedon pattern and by navigating to open frontiers within the region. In the second stage, the robot revisits obstacles and follows their boundaries. This ensures all space is covered and no area is missed.

We compared our method to five off-line methods that access an *a priori* map of the environment and have perfect sensing and localization. On average, our approach produces

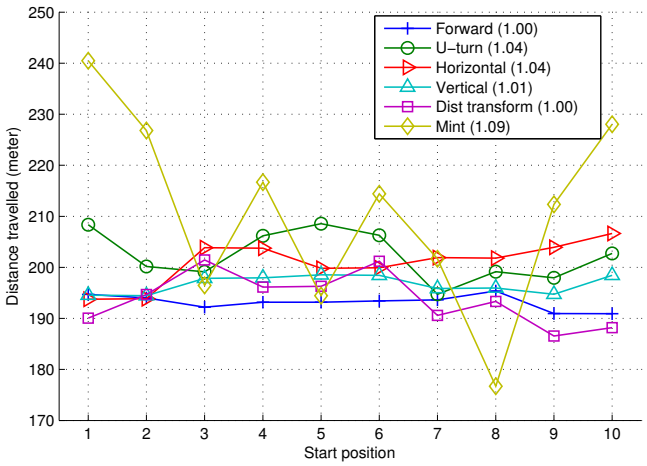


Fig. 5. Travel distance over all 10 start poses in IEC room for Mint and the 5 off-line methods. Normalized path lengths in parentheses.

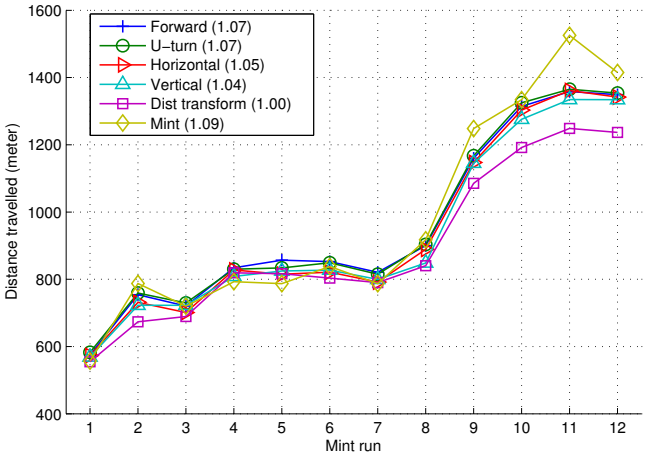


Fig. 7. Travel distance over all 12 Mint runs with factor in parentheses.

trajectories that are only about 9 % longer than the distance transformation from Zelinsky *et al.* [1]. Although this comparison is not apples-to-apples it provides a good indication of the performance of our method.

As a future work, we would like to address the problem of coverage when the underlying occupancy grid is not a global map but composed of multiple sub-maps anchored to pose nodes of a graph in a SLAM system.

Acknowledgment

The authors would like to thank Mike Stout, Gabe Brisson, and Jacob Madden for implementing an earlier version of the proposed method for systematic floor coverage.

References

[1] A. Zelinsky, R. Jarvis, J. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," *Int. Conference on Advanced Robotics (ICAR)*, pp. 533–538, 1993.

[2] J.-S. Gutmann, K. Culp, M. Munich, and P. Pirjanian, "The social impact of a systematic floor cleaner," in *Workshop on Advanced Robotics and its Social Impacts (ARSO)*, Munich, Germany, May 2012.

[3] T. Palleja, M. Tresanchez, M. Teixido, and J. Palacin, "Modeling floor-cleaning coverage performances of some domestic mobile robots in a reduced scenario," *Robotics and Autonomous Systems*, vol. 58, no. 1, pp. 37–45, 2010.

[4] S. C. Wong, L. Middleton, B. A. MacDonald, and N. Auckland, "Performance metrics for robot coverage tasks," in *Australasian Conference on Robotics and Automation*, 2002.

[5] *Vacuum cleaners for household use*, International Electrotechnical Commission Part 3: Cleaning robots for household use – Dry cleaning – Methods of measuring performance 60 312-3, 2013, to appear.

[6] I. Ulrich, F. Mondada, and J.-D. Nicoud, "Autonomous vacuum cleaner," *Robotics and Autonomous Systems*, vol. 19, no. 3, 1997.

[7] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry*, vol. 17, no. 1, pp. 25–50, 2000.

[8] B. Yamauchi, A. Schultz, and W. Adams, "Mobile robot exploration and map-building with continuous localization," in *Int. Conference on Robotics and Automation (ICRA)*, 1998, pp. 3715–3720.

[9] C. Stachniss, *Robotic mapping and exploration*, ser. Springer Tracts in Advanced Robotics. Springer, 2009, vol. 55.

[10] I. Rekleitis, "Simultaneous localization and uncertainty reduction on maps (SLURM): Ear based exploration," in *Int. Conference on Robotics and Biomimetics (ROBIO)*, 2012, pp. 501–507.

[11] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, 2001.

[12] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Int. Conference on Field and Service Robotics*, 1997.

[13] H. Choset, "Coverage of known spaces: the boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, no. 3, pp. 247–253, 2000.

[14] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Int. Conf. on Robotics and Automation (ICRA)*, 2001.

[15] E. U. Acar and H. Choset, "Robust sensor-based coverage of unstructured environments," in *Int. Conference on Robotics and Intelligent Systems (IROS)*, 2001, pp. 61–68.

[16] E. U. Acar, H. Choset, and J. Y. Lee, "Sensor-based coverage with extended range detectors," *Transactions on Robotics*, vol. 22, no. 1, pp. 189–198, 2006.

[17] E. Garcia and P. Gonzalez de Santos, "Mobile-robot navigation with complete coverage of unstructured environments," *Robotics and Autonomous Systems*, vol. 46, no. 4, pp. 195–204, 2004.

[18] S. C. Wong and B. A. MacDonald, "A topological coverage algorithm for mobile robots," in *Int. Conference on Robotics and Intelligent Systems (IROS)*, 2003, pp. 1685–1690.

[19] J. W. Kang, S. J. Kim, M. J. Chung, H. Myung, J. H. Park, and S. W. Bang, "Path planning for complete and efficient coverage operation of mobile robots," in *Int. Conference on Mechatronics and Automation (ICMA)*, 2007, pp. 2126–2131.

[20] H. Myung, H.-m. Jeon, W.-Y. Jeong, and S.-W. Bang, "Virtual door-based coverage path planning for mobile robot," *Advances in Robotics*, pp. 197–207, 2009.

[21] H. H. Viet, V.-H. Dang, M. N. U. Laskar, and T. Chung, "BA*: an online complete coverage algorithm for cleaning robots," *Applied Intelligence*, pp. 1–19, 2012.

[22] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," in *Int. Conference on Robotics and Automation (ICRA)*, 2001, pp. 1927–1933.

[23] ——, "Spiral-STC: An on-line coverage algorithm of grid environments by a mobile robot," in *Int. Conference on Robotics and Automation (ICRA)*, 2002, pp. 954–960.

[24] R. Mannadiar and I. Rekleitis, "Optimal coverage of a known arbitrary environment," in *Int. Conference on Robotics and Automation (ICRA)*, 2010, pp. 5525–5530.

[25] A. Xu, C. Viriyasuthee, and I. Rekleitis, "Optimal complete terrain coverage using an unmanned aerial vehicle," in *Int. Conference on Robotics and Automation (ICRA)*, 2011, pp. 2513–2519.

[26] C. Das, A. Becker, and T. Bretl, "Probably approximately correct coverage for robots with uncertainty," in *Int. Conference on Robotics and Intelligent Systems (IROS)*, 2011, pp. 1160–1166.

[27] J.-S. Gutmann, E. Eade, P. Fong, and M. Munich, "Vector field SLAM – localization by learning the spatial variation of continuous signals," *Transactions on Robotics*, vol. 28, no. 3, pp. 650–667, 2012.

[28] J.-S. Gutmann, D. Goel, and M. Munich, "Scaling vector field SLAM to large environments," *Intelligent Autonomous System (IAS)*, vol. 12, pp. 89–100, 2013.

[29] E. Eade, P. Fong, and M. E. Munich, "Monocular graph SLAM with complexity reduction," in *Int. Conference on Robotics and Intelligent Systems (IROS)*, Taipei, Taiwan, October 2010, pp. 3017–3024.

[30] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[31] J.-S. Gutmann, P. Fong, L. Chiu, and M. Munich, "Challenges of designing a low-cost indoor localization system using active beacons," in *Int. Conference on Technologies for Practical Robot Applications (TePRA)*, 2013.