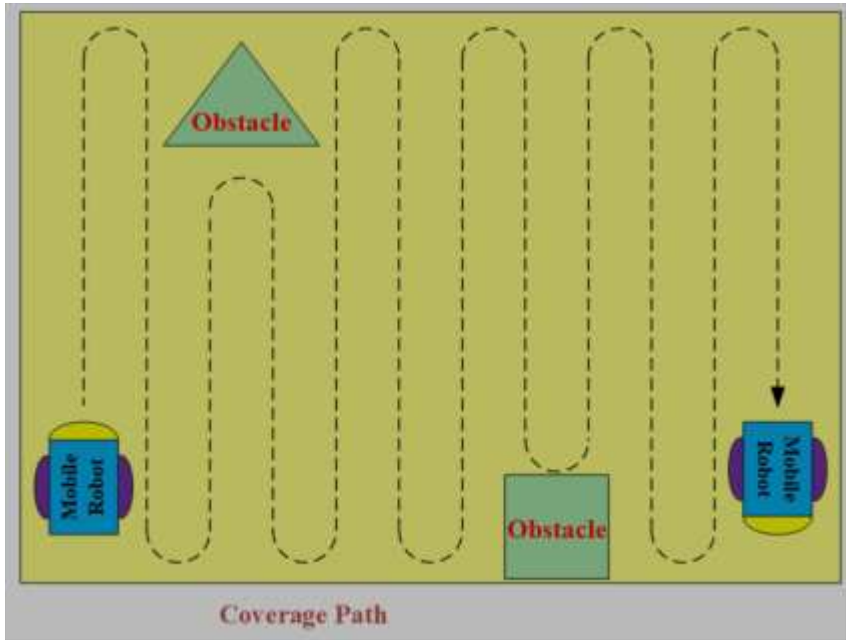
Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation

OSAMA MUHAMMAD

SHOAIB HASAN

GENERAL INFORMATION

- Authored by Tae-Kyeong Lee, Sang-Hoon Baek, Young-Ho Choi, Se-Young Ho
- Pohang University of Science and Technology, Pohang, South Korea
- Published in 2011



OVERVIEW - GOAL

- Reduce the time and energy required to cover an area
- Mobile robots like robot vacuums, autonomous lawn mowing, etc
- Aims to improve energy efficiency at a reduced cost, while improving coverage

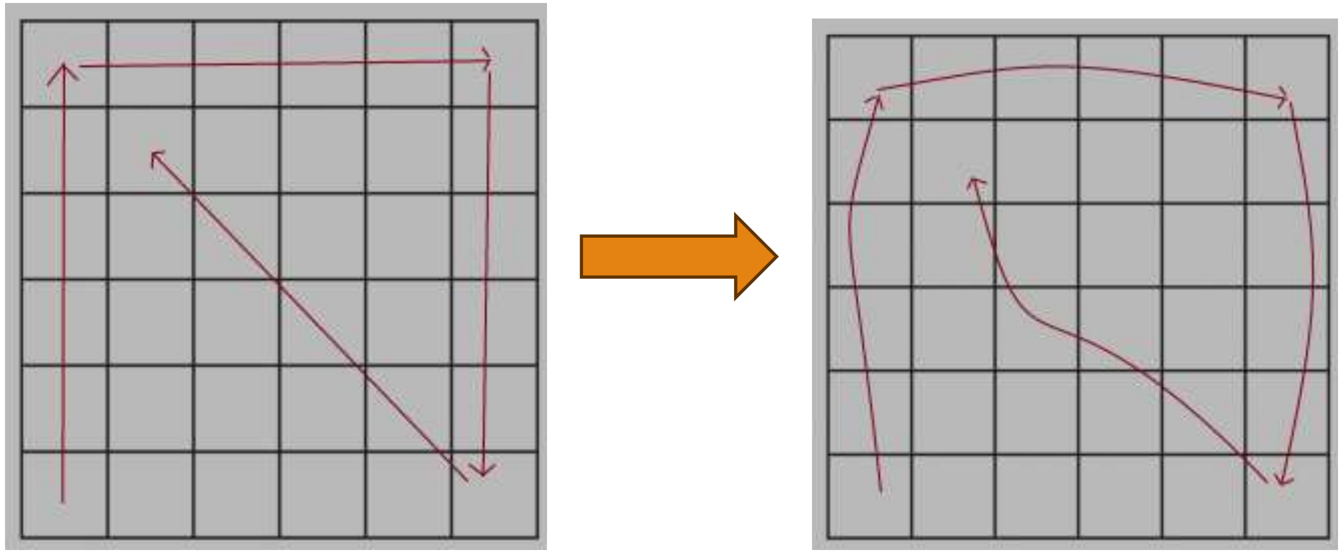
OVERVIEW - METHOD

- **Reduce path overlap** or **Increase average velocity**

1. Map out space into a high-resolution grid map

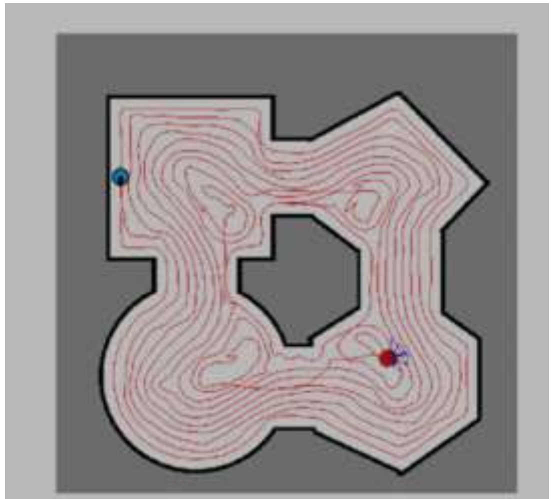
2. Plan a path and avoid obstacles

3. Improve the path by adding curves

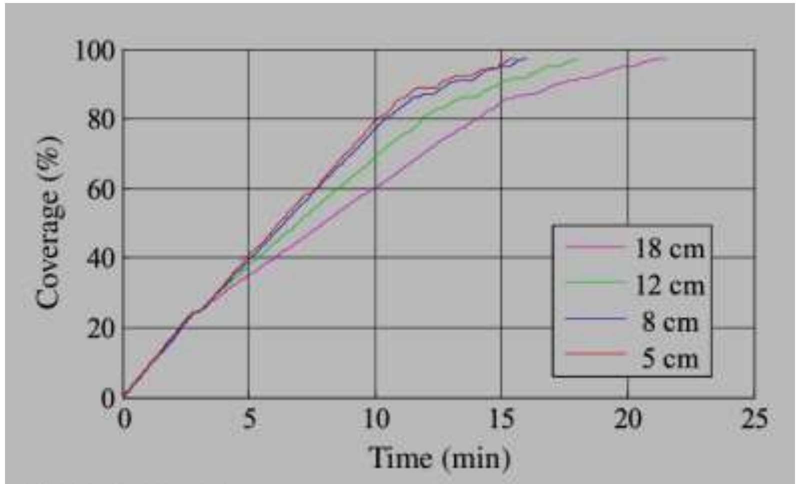


OVERVIEW - RESULTS

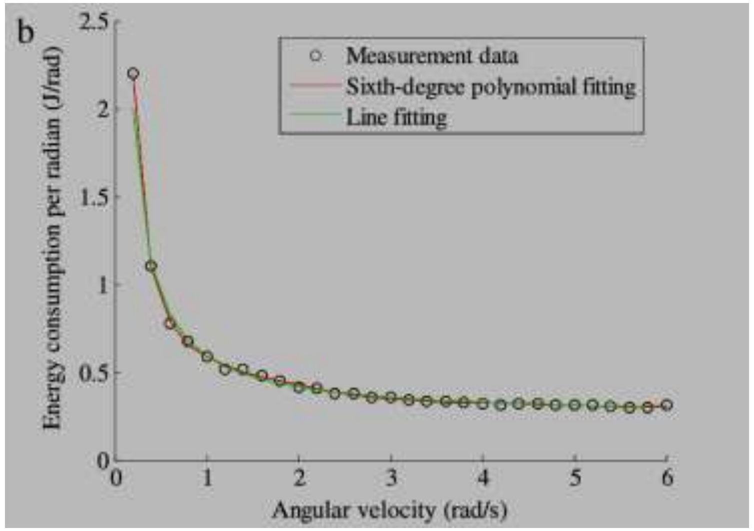
- Algorithm proposed is simulated in environments with irregular shaped obstacles
- Coverage time, average velocity and energy efficiency of the robot is monitored
- Robots perform better than other algorithms like Spiral-STC



(a) 18 cm.



(e) Coverage graph.



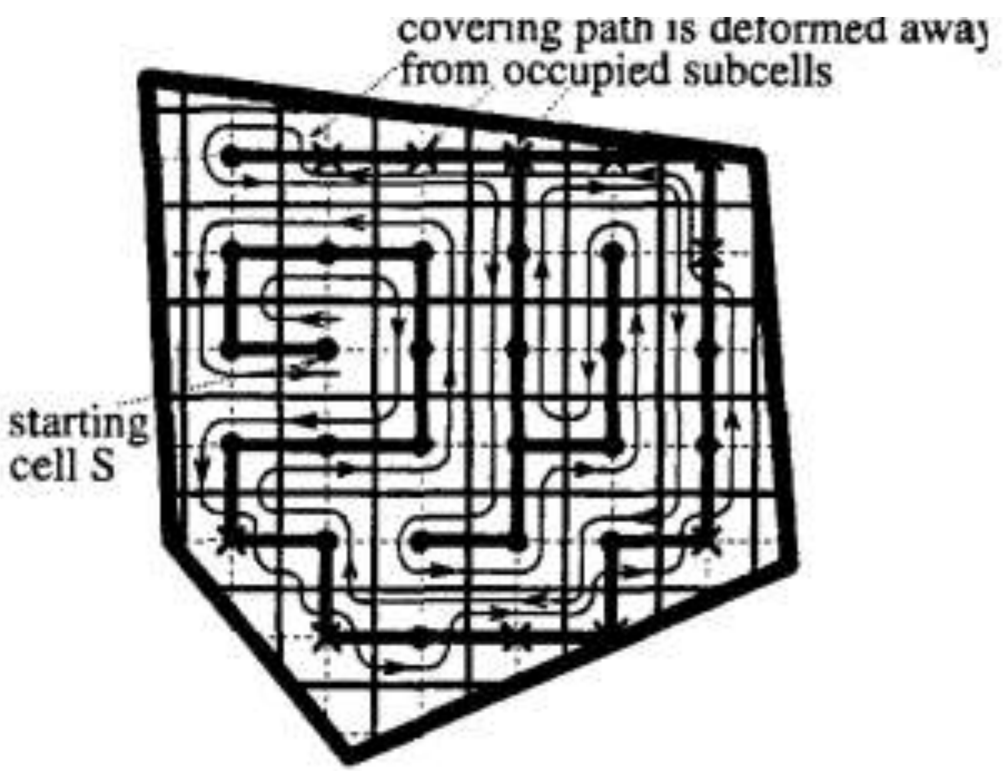


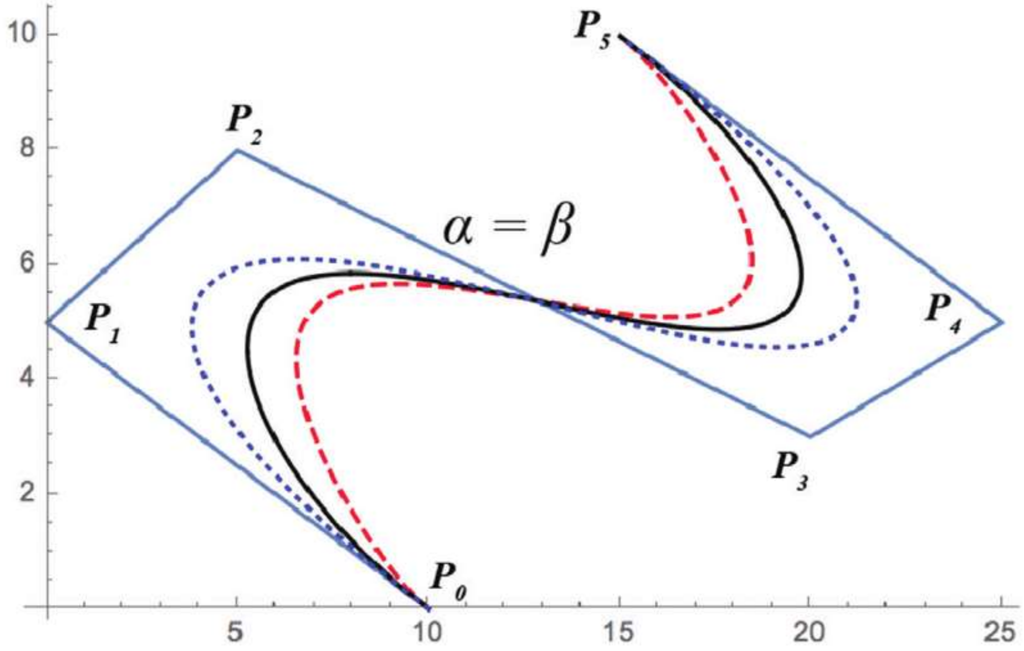
Figure 5: An execution example of the full *Spiral-STC* algorithm.

RELATED WORKS – SPIRAL STC

- Discretizes the work-area into a grid and follows a path around a spanning tree
- Treats partially occupied cells as special nodes for repetitive coverage
- Aims to provide full coverage of an area and revisits cells multiple times in the process

RELATED WORKS – BEZIER CURVE

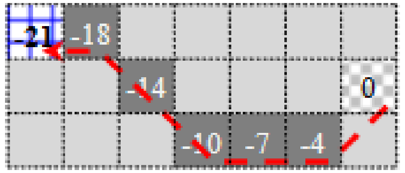
- Parametric curves used to model smooth and scalable shapes
- Defined by a set of control points
- Represented mathematically with a parametric function Q(t)



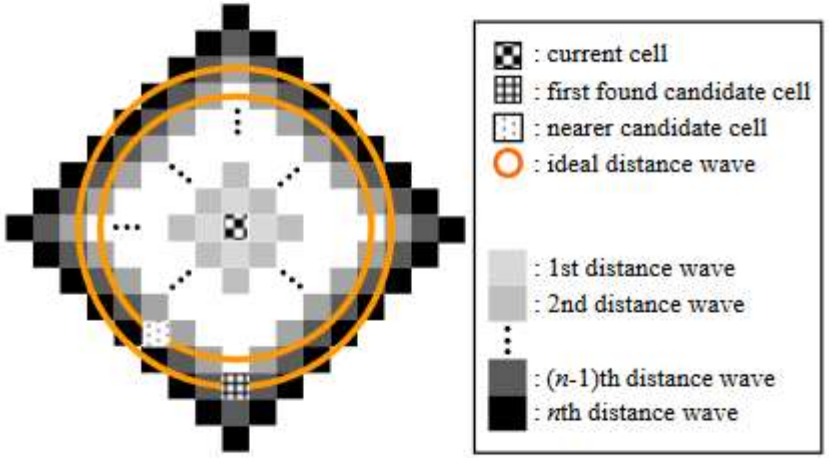
Constrained inverse distance transform (CIDT)

- To help the robot find and get to the nearest target cell after finishing a spiral path, an algorithm is needed

- In a simple inverse distance transform (IDT), distance propagation is used from a start cell and ends at an unknown candidate cell, highlighting a shortest path.
  - A full scan (in the shape of squares) must be completed, even if the nearest target cell is found first
  - Minimum distance cost of nearest target cell ≤ Minimum distance cost of current wave
  - Placing this constraint saves memory and time, hence CIDT



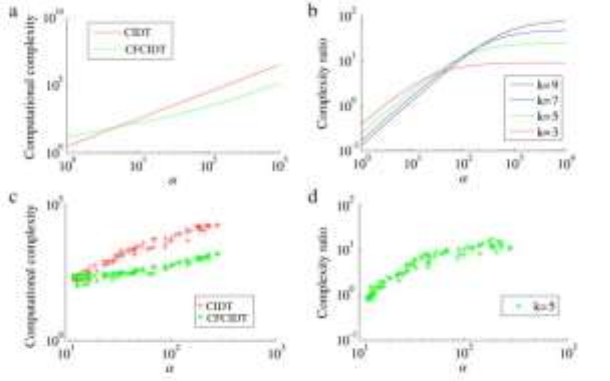
(c) Gradient descent search from the current cell to the target cell after reversing the sign of the cells on the path

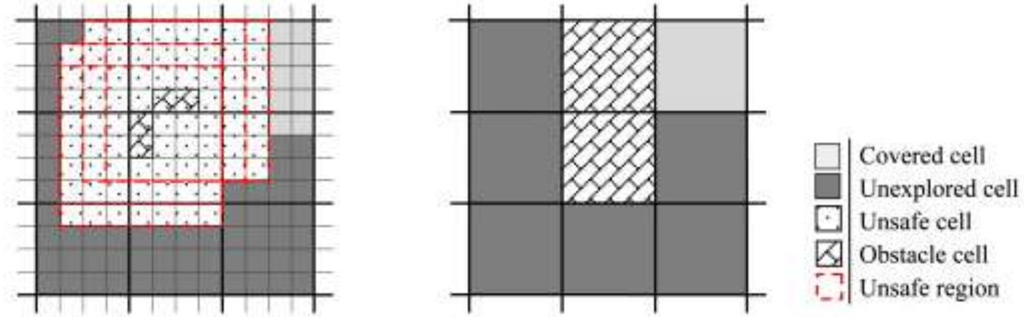


Coarse-to-fine constrained inverse distance transform (CFCIDT)

- The coverage map from the path algorithms are used as a configuration map by enlarging the obstacle cells.
- A block map is also made with the same occupancy as the config. map
- Course CIDT is applied to the block map to find the nearest unexplored block.
- Some of the points near the target are augmented to help with finding a good path. Once the path is found, a finer CIDT is applied to the configuration map
- After finding the nearest unexplored cell, it's used as a seed point for sampling virtual wall points
- Using a gradient descent search, the robot follows the path to the target cell, using directions from the virtual wall path.

CFCIDT has a lower computational complexity than CIDT, even with configuration and block map generation, which can be done incrementally during coverage.



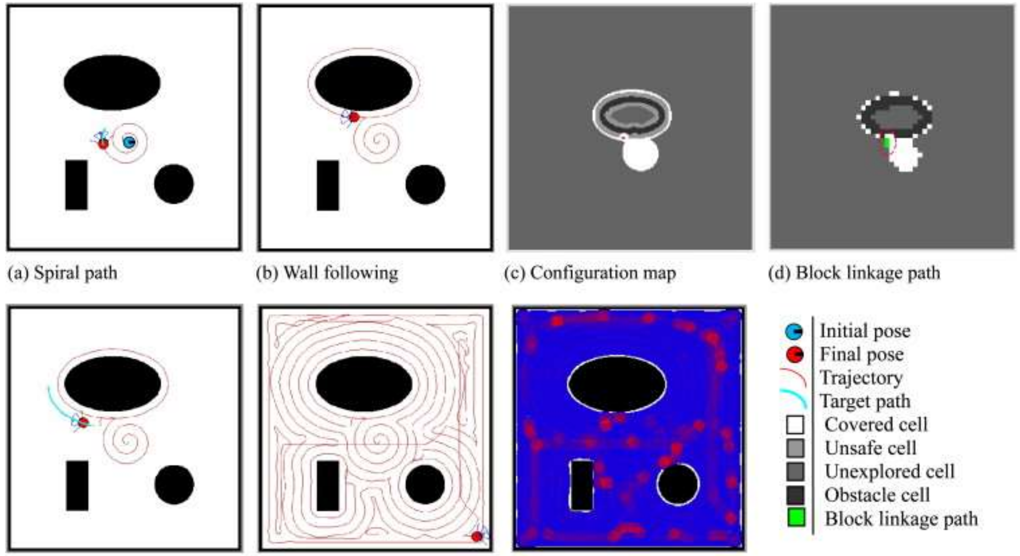


(a) Configuration map with $r = 3$

(b) Block map built from (a) with $k = 4$

ALGORITHM BREAKDOWN

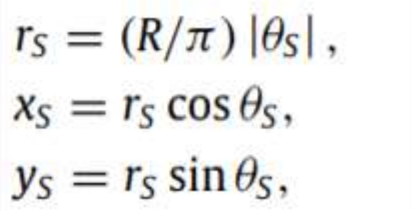
1. Spiral Path until an obstacle is detected
2. Wall following behaviour is executed until the start point of the path is reached
3. Robot searches for any unexplored cells using coarse-to-fine constrained inverse distance transform (CFCIDT)
4. If there are unexplored cells present, the robot moves to the nearest unexplored cell and performs virtual wall tracking
5. If an obstacle is detected while following the virtual path, the robot will execute wall following behavior
6. Steps 2 to 5 are executed until the CFCIDT returns no unexplored cells


(a) Spiral path (b) Wall following (c) Configuration map (d) Block linkage path

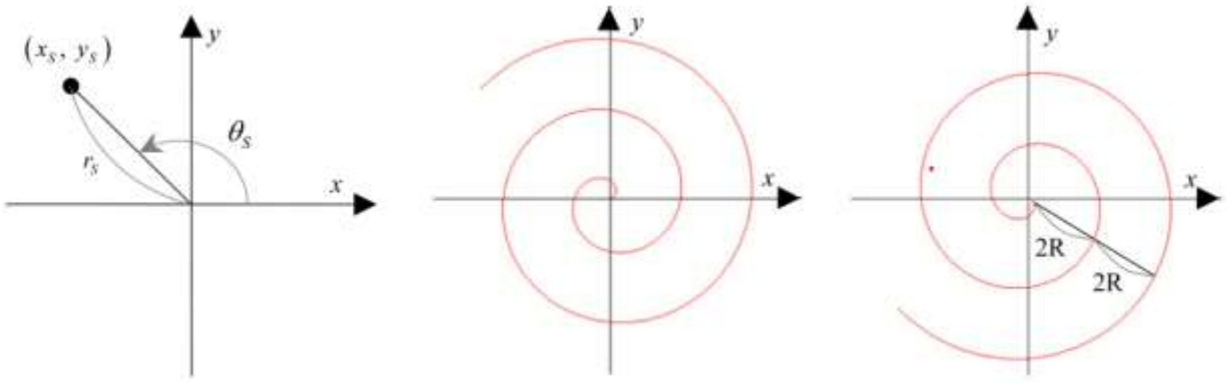
(e) Virtual wall path (f) Entire trajectory (g) Coverage

ALGORITHM – SPIRAL PATH

- Spiral Path is the initial behaviour of the robot
- Starting point of the robot is defined as the origin in 2D plane
- X-axis is defined in the direction the robot is facing



$$r_S = (R/\pi)\,|\theta_S|\,,$$
$$x_S = r_S \cos\theta_S,$$
$$y_S = r_S \sin\theta_S,$$



(a) Parameters for the spiral path.
(b) Spiral with increasing θ_S.
(c) Spiral with decreasing θ_S.

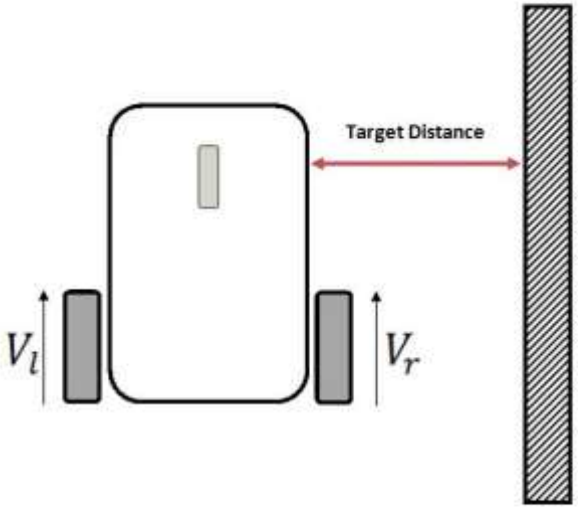
R = Radius of the coverage tool

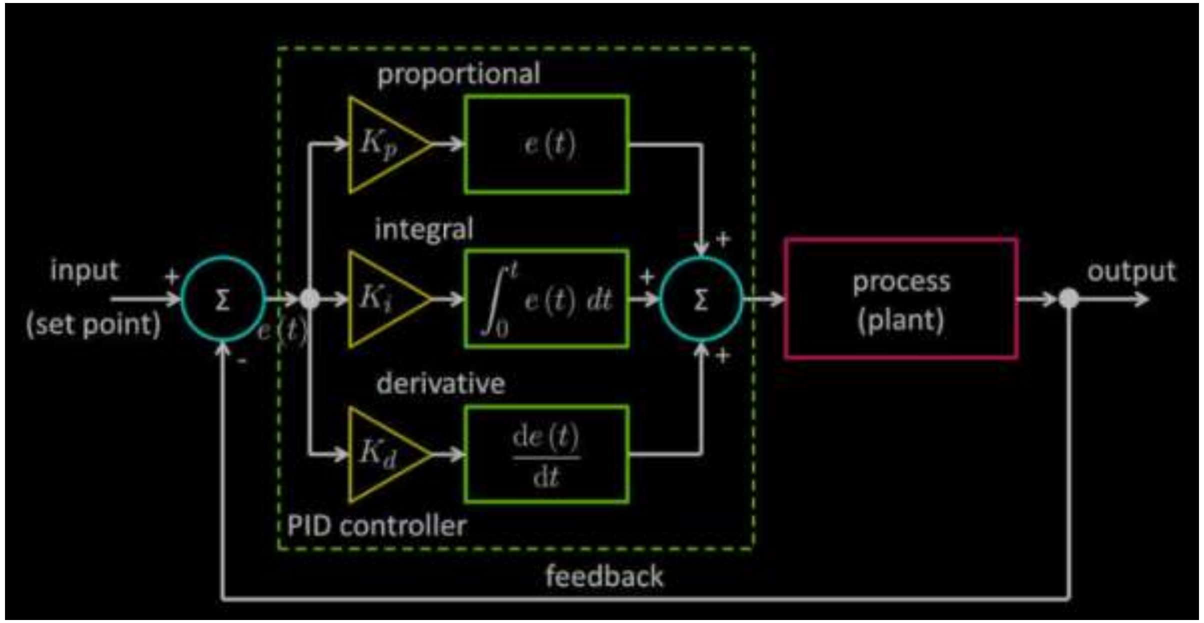
Xs / Ys = Location of the point

Ө = Accumulated rotation angle of the point

ALGORITHIM – WALL FOLLOWING PATH

- Determine distance between obstacle and robot using range sensor measurements
- Use PID Controller to achieve the required distance from the wall
- Account for physical limitations from robot body and detection range of sensors





ALGORITHIM – VIRTUAL WALL PATH

- Virtual wall path is aligned to the virtual wall boundary that is created around the area the robot has already visited

- Modelling the virtual wall path contains following steps:

  I. Sampling Wall Points

  II. Sampling Path Points

  III. Filtering Path Points

  IV. Optimization path using Particle Swarm Optimization

ALGORITHIM – VIRTUAL WALL PATH CONT.

➢ Sampling Wall Points

- Task to determine the location of the wall
- Points are the last covered cells before unexplored cells
- First point is provided by the CFCIDT

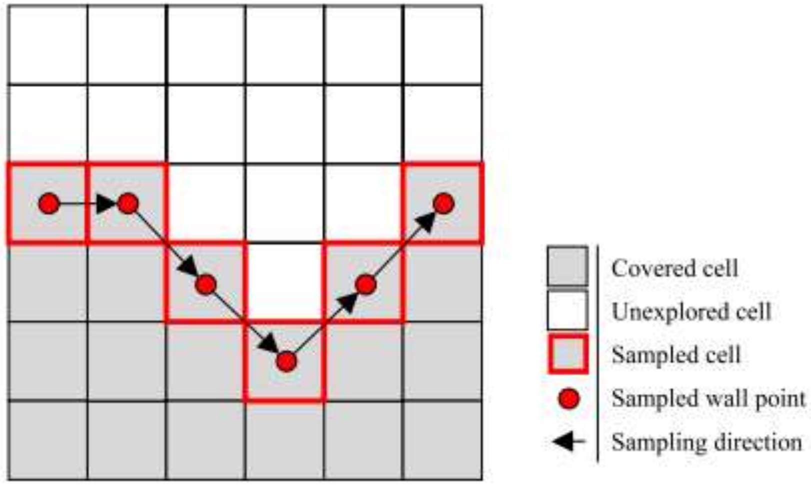


Fig. 3. Wall point sampling with the right-hand rule.

➢ Sampling Path Points

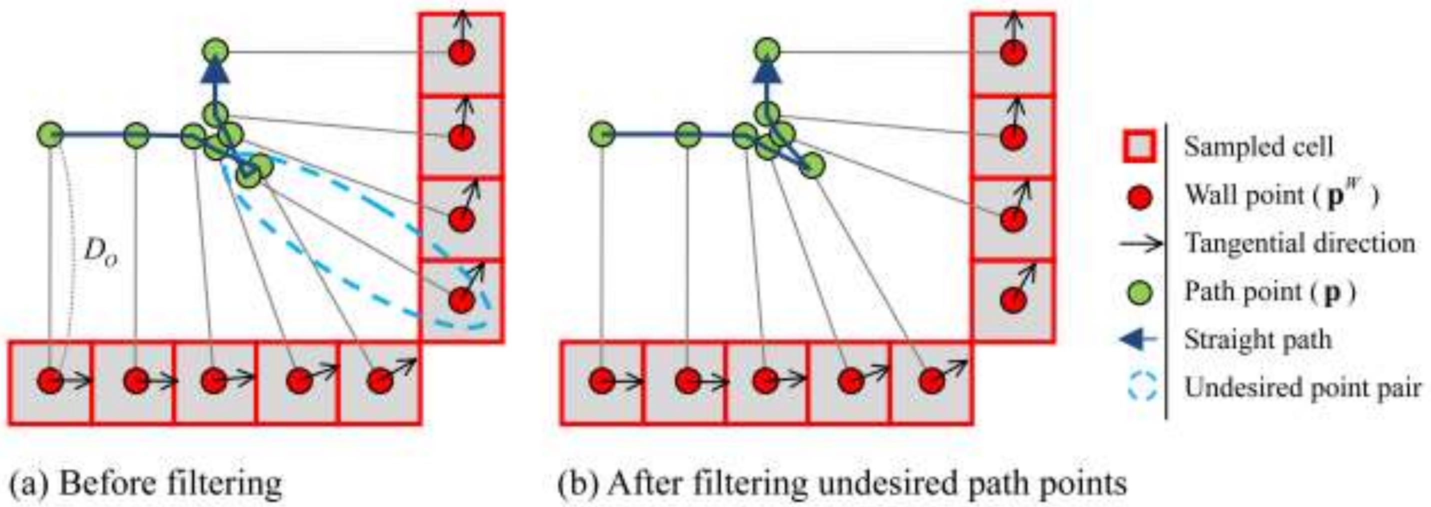
- Task to determine the tangential direction of each wall point
- Assume all sampled points are on smooth curve r(s)
- Determining the derivative for that curve at different points provides the tangential direction at our sampled points

$$\theta_k^W = a\tan 2\left(r_y'(0), r_x'(0)\right),$$

- Based on the tool width and grid cell width, a curve-path offset from the wall points is generated

ALGORITHIM – VIRTUAL WALL PATH CONT.

- Filtering Path Points
  - Task to filter out undesired path points
  - Two consecutive path points are chosen, and each path point is connected to its corresponding wall point
  - If the lines intersect, one of the path points is removed



(a) Before filtering

(b) After filtering undesired path points

Fig. 4. Filtering of undesired path points.

ALGORITHIM – VIRTUAL WALL PATH CONT.

Filtering Path Points Cont.

- Tougher to formulate a smooth path with unnecessary points
- Minimal set of points is determined that covers all wall points
- Quintic Bezier Splines are interpolated from the filtered path points

$$\mathbf{Q}_i(u_i) = (1-u_i)^5\,\mathbf{P}_{i,0} + 5(1-u_i)^4\,u_i\mathbf{P}_{i,1} + 10(1-u_i)^3\,u_i^2\mathbf{P}_{i,2} + 10(1-u_i)^2\,u_i^3\mathbf{P}_{i,3} + 5(1-u_i)\,u_i^4\mathbf{P}_{i,4} + u_i^5\mathbf{P}_{i,5}, \quad (13)$$

$$\begin{cases} \mathbf{P}_{i,0} = \hat{\mathbf{p}}_i, \\ \mathbf{P}_{i,5} = \hat{\mathbf{p}}_{i+1}, \\ \mathbf{P}_{i,1} = \mathbf{P}_{i,0} + \frac{1}{5}\mathbf{T}_i, \\ \mathbf{P}_{i,4} = \mathbf{P}_{i,5} - \frac{1}{5}\mathbf{T}_{i+1}, \\ \mathbf{P}_{i,2} = \frac{1}{20}\mathbf{A}_i + 2\mathbf{P}_{i,1} - \mathbf{P}_{i,0}, \\ \mathbf{P}_{i,3} = \frac{1}{20}\mathbf{A}_{i+1} + 2\mathbf{P}_{i,4} - \mathbf{P}_{i,5}, \end{cases}$$

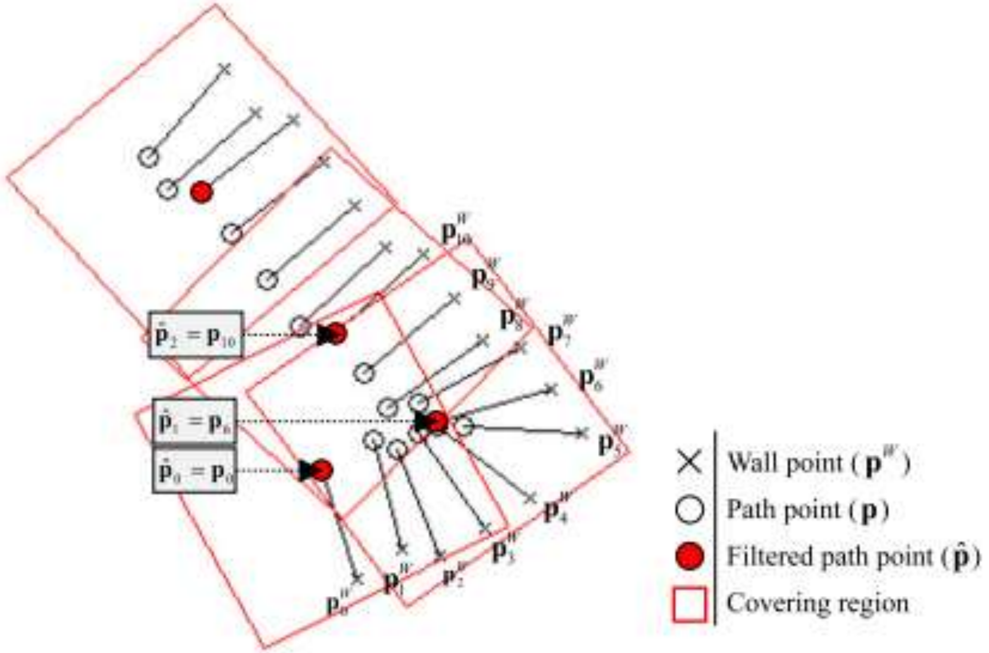
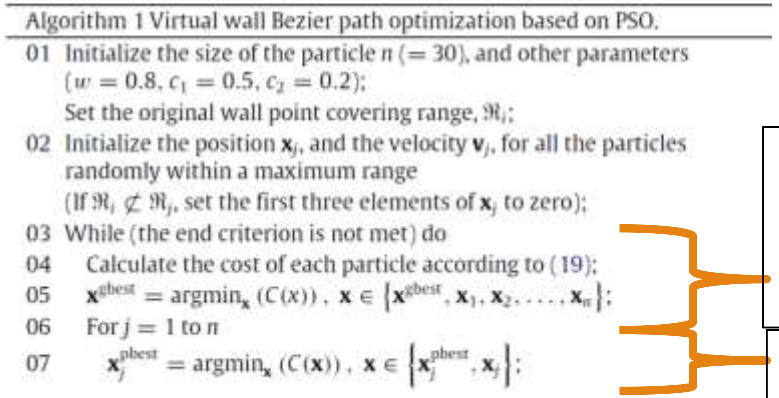

$\hat{\mathbf{p}}_2 = \mathbf{p}_{10}$
$\hat{\mathbf{p}}_1 = \mathbf{p}_6$
$\hat{\mathbf{p}}_0 = \mathbf{p}_0$
$\mathbf{p}''_{10}$
$\mathbf{p}''_9$
$\mathbf{p}''_8$
$\mathbf{p}''_7$
$\mathbf{p}''_6$
$\mathbf{p}''_5$
$\mathbf{p}''_4$
$\mathbf{p}''_3$
$\mathbf{p}''_2$
$\mathbf{p}''_1$
$\mathbf{p}''_0$
× Wall point ($\mathbf{p}''$)
○ Path point ($\mathbf{p}$)
● Filtered path point ($\hat{\mathbf{p}}$)
□ Covering region

Fig. 5. Minimum set of path points covering the entire wall points. The wall point covering range and connection gap of a filtered path point $\hat{\mathbf{p}}_0$ are equal to $\Re_0 = \{0, 1, 2, 3\}$ and $C_0 = \sum_{n=0}^{5} |\overline{\mathbf{p}_n\mathbf{p}_{n+1}}|$.

ALGORITHIM – OPTIMIZATION USING PSO

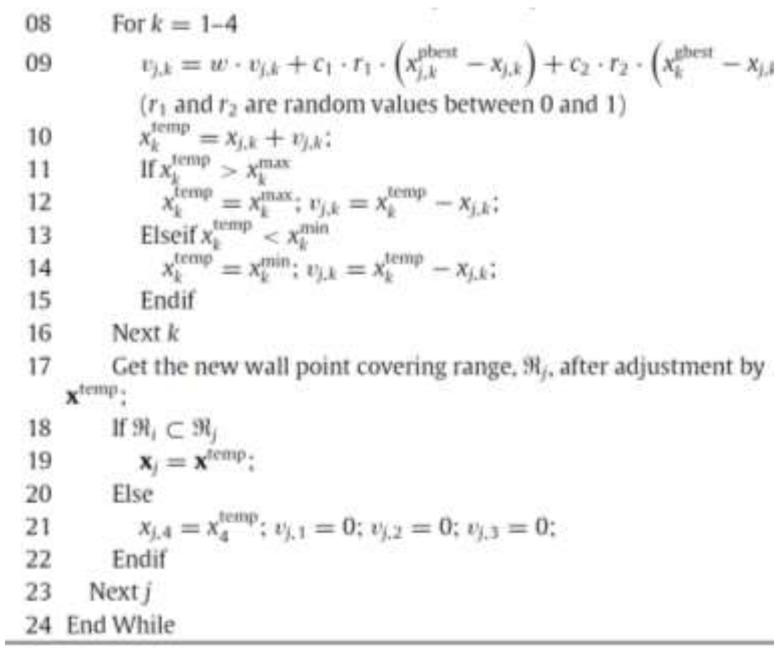


Algorithm 1 Virtual wall Bezier path optimization based on PSO.

```
Initialize the size of the particle $n$ (= 30), and other parameters
($w = 0.8, c_1 = 0.5, c_2 = 0.2$);
Set the original wall point covering range, $\Re_i$;
Initialize the position $\mathbf{x}_j$, and the velocity $\mathbf{v}_j$, for all the particles
randomly within a maximum range
(If $\Re_i \not\subset \Re_j$, set the first three elements of $\mathbf{x}_j$ to zero);
While (the end criterion is not met) do
  Calculate the cost of each particle according to (19);
  $\mathbf{x}^{gbest} = \mathrm{argmin}_{\mathbf{x}} (C(x)), \mathbf{x} \in \{\mathbf{x}^{gbest}, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$;
  For $j = 1$ to $n$
    $\mathbf{x}_j^{pbest} = \mathrm{argmin}_{\mathbf{x}} (C(\mathbf{x})), \mathbf{x} \in \{\mathbf{x}_j^{pbest}, \mathbf{x}_j\}$;
```

Determine the cost for each point and the global best between all points

Find the personal best position for each point



```
    For $k$ = 1–4
      $v_{j,k} = w \cdot v_{j,k} + c_1 \cdot r_1 \cdot \left(x_{j,k}^{pbest} - x_{j,k}\right) + c_2 \cdot r_2 \cdot \left(x_k^{gbest} - x_{j,k}$
      ($r_1$ and $r_2$ are random values between 0 and 1)
      $x_k^{temp} = x_{j,k} + v_{j,k}$;
      If $x_k^{temp} > x_k^{max}$
        $x_k^{temp} = x_k^{max}$; $v_{j,k} = x_k^{temp} - x_{j,k}$;
      Elseif $x_k^{temp} < x_k^{min}$
        $x_k^{temp} = x_k^{min}$; $v_{j,k} = x_k^{temp} - x_{j,k}$;
      Endif
    Next $k$
    Get the new wall point covering range, $\Re_j$, after adjustment by
  $\mathbf{x}^{temp}$;
    If $\Re_i \subset \Re_j$
      $\mathbf{x}_j = \mathbf{x}^{temp}$;
    Else
      $x_{j,4} = x_4^{temp}$; $v_{j,1} = 0$; $v_{j,2} = 0$; $v_{j,3} = 0$;
    Endif
  Next $j$
End While
```

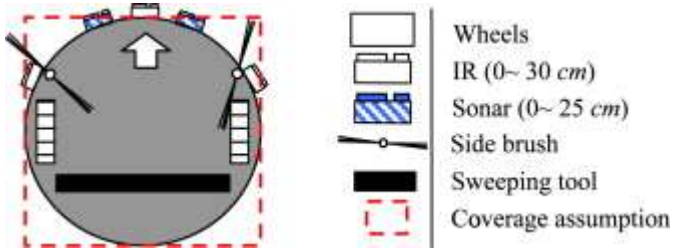
Update the points velocity based on its personal and global best positions.

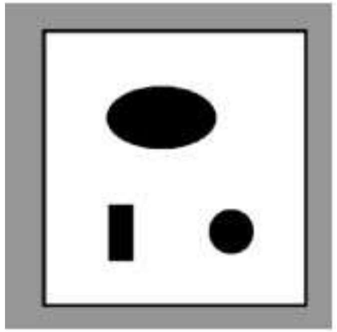
Update the points position based on its velocity and limit it to a certain range

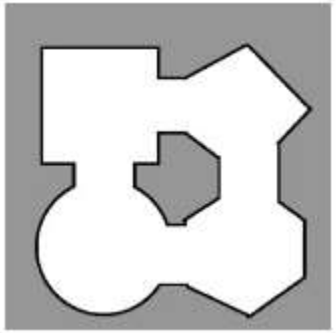
Update the position of the point only if its range is within the new range

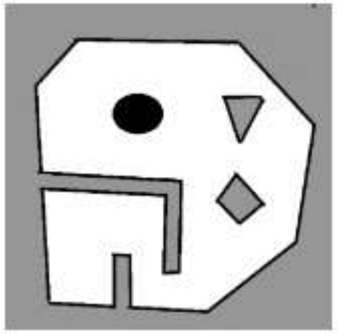
Simulation Setup

- Done on Mobile Robot Coverage Simulator (MRCS)
- Robot modelled after a floor cleaning robot platform
- Developed power models for the wheels of the robot
- 3 different test environments used to test coverage



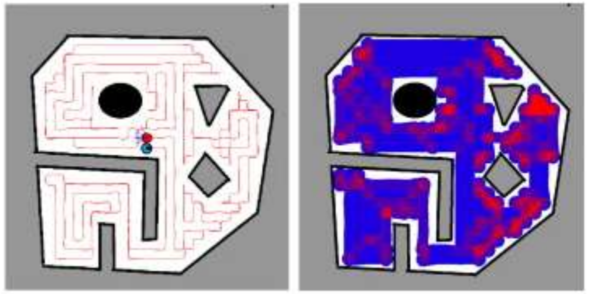




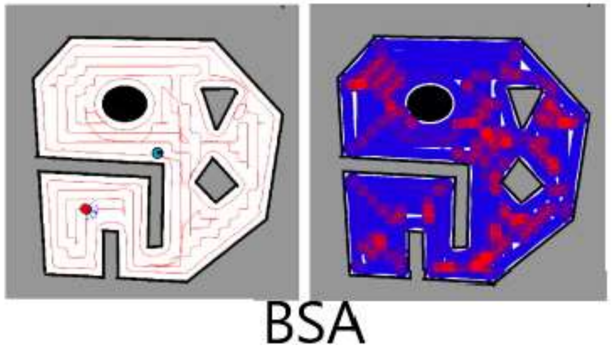


Performance comparison

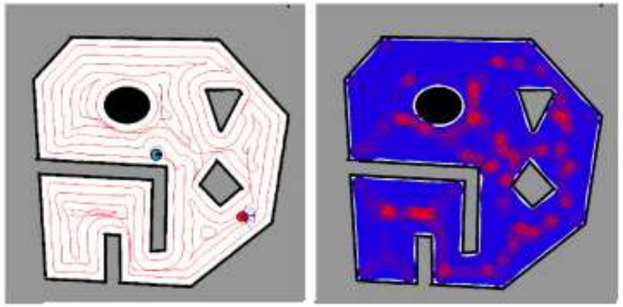
- Comparison against spiral-STC and backtracking spiral (BSA) algorithms
- Performance metrics were coverage, normalized time to coverage, average velocity, and energy efficiency (to complete a covered area)



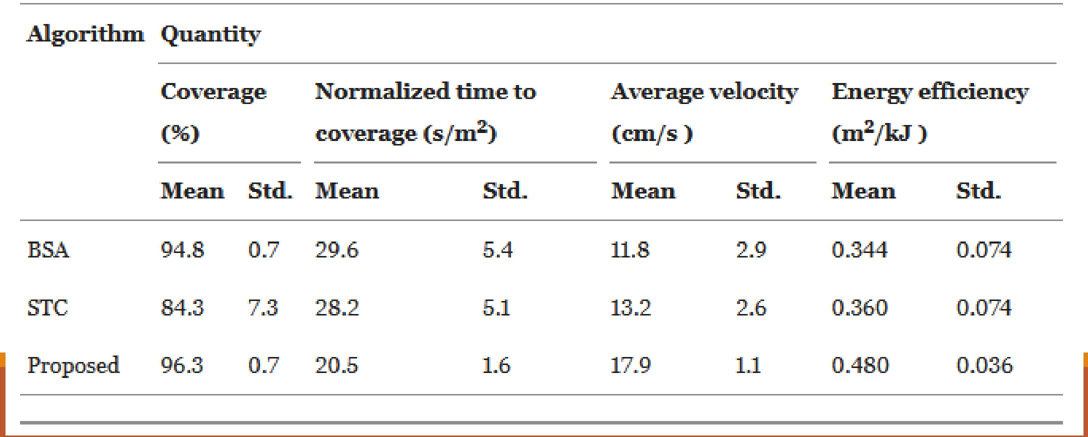
Spiral



BSA



Proposed Algorithm



Algorithm	Quantity							
	Coverage (%)		Normalized time to coverage (s/m^2)		Average velocity (cm/s)		Energy efficiency (m^2/kJ)	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
BSA	94.8	0.7	29.6	5.4	11.8	2.9	0.344	0.074
STC	84.3	7.3	28.2	5.1	13.2	2.6	0.360	0.074
Proposed	96.3	0.7	20.5	1.6	17.9	1.1	0.480	0.036

Conclusion

- Future works: Course to fine strategy used in other applications, such as wayfinding strategy for robot navigation, and hybrid 3D mapping systems.

- Pros:
  - Relatively organized and concise
  - Good use of legends, visuals, and detailed comparisons
  - Relevance to increasing demand of mobile robots in commercial and industrial workspaces

- Cons:
  - Some lack of explanation of certain topics, like kinodynamic constraints, and Bezier splines.
  - Doesn't address localization problems

👍 Content

👍 Clarity

👍 Results

👍 Organization

Thank you!