A Triangulation-Based Coverage Path Planning

Vatana An, *Member, IEEE*, Zhihua Qu, *Fellow, IEEE*, Frank Crosby, *Member, IEEE*, Rodney Roberts, *Senior Member, IEEE*, and Vithia An, *Member, IEEE*

***Abstract*—In this paper, we present an approach to local coverage path (CP) planning for a circular mobile sensor. Our family of algorithms provides completed and overlapped coverage control, curvature and path length control, localization, choice of observer's placement based on sensing range, and first-order differentiable CP. Our family of algorithms begins by partitioning the target region (TR) into several regular triangulations (RTs). Based on the size of the RT and the sensing range, the number and location of observers are determined for all RT. All observers found are then used as waypoints (WPs) to generate baseline CP with the Traveling Salesman Problem's nearest neighbor algorithm. The proposed algorithm solved the problems of how to design a differentiable and collision-free CP for all sensing range, limited and sufficient, while providing completed coverage control, overlapped coverage control, curvature and path length control, localization, collision avoidance, and the choice of observer's placement based on sensing range and the environment. The main technical contributions of the proposed approach is to provide a holistic solution that segments any TR, uses triangulation to determine the observation WPs and then compute the smooth and collision-free CP. Computer simulations are provided to validate the effectiveness and correctness of our algorithms.**

***Index Terms*—Half-plane, internal and external tangent, nonholonomic, triangulation, visible polygon (VP).**

I. Introduction

COVERAGE path (CP) planning for a mobile robot to find motion path that observe the entire work region receive numerous interests from researchers in the last 15 years [1]–[13]. Examples of CP planning include automate floor scrubbing, automated wall painting, vacuum cleaning, and security patrolling. Recent interests provide motivation to pursue high performance and scalable algorithms in CP planning. Maximizing coverage performance is the most challenging issue for robotic cleaner [3]. Reference [3] suggested field programmable gate arrays' and finite state machines' approaches to cleaning path with the cleaning CP being spiral, zigzag, or random path. Reference [11] considers a smooth CP for a vacuum cleaner in a rectilinear environment and the path design is geared toward minimizing human's energy.

Cellular decompositions are popular techniques in CP planning [5]. All cellular decomposition techniques divide the work area into "simple" cells. Reference [7] identified the flexibility of CP planning with the triangular-cell-based map technique which has more accessible direction than the rectangular cell-based decomposition. This paper considered all triangular cells to be of identical size yet the resulting CP is discontinuous which is inefficient in term of energy consumption.

Reference [13] considers approximate solutions to visibility problem with infinite sensing range and the nearly optimal CP is obtained in several thousand iterations.

Our technique considered all sensing ranges and it can be applied to vacuum cleaning application by setting the sensing range equal to the robot's platform. When the sensing range is greater than the robot platform, the same algorithm becomes useful for security patrolling or search operation. Our algorithm only requires a few iterations to obtain near optimal CP based on the range of the sensor. In addition, our algorithm does not constraint the triangular cell to be of identical size or shape. Table I lists common acronyms that will be used throughout this paper. Definitions for each of the terms in this table will be given when they are first introduced.

II. Problem Formulation

This section presents some assumptions and definitions required to solve the problem introduced in Section I.

Assumption 1: The robot being studied is a two-wheeled robot, enveloped by a two-dimensional circle, with the center at $O(t) = (x, y)$ and of radius R_r. Its motion obeys a nonholonomic constraint with velocity vector expressed as $v_r(t)$. Note that the position and velocity are a function of time because the robot is continuously moving.

Assumption 2: The radius or range of robot's motion sensor is R_s. R_s is greater than R_r. Fig. 1(b) illustrates the sensor maximum square (SMS). Since the sensor's diameter is $2R_s$, sensor's maximum square dimension is $\text{SMSD} = \sqrt{2}R_s$.

Assumption 3: The static objects are denoted by the symbol O_i, where the subscript $i = 1, \ldots, n$ represent the obstacle

Manuscript received October 15, 2017; accepted February 4, 2018. This work was supported by the U.S. Navy's Internal Laboratory Independence Research. This paper was presented in part in 3rd Australian Control Conference (2013), Perth, Australia, and in part in Robotic and Sensors Environments (2013), Washington, DC, USA. This paper was recommended by Associate Editor S. Nahavandi. *(Corresponding author: Vatana An.)*

V. An and F. Crosby are with the Science and Technology Department, United States Navy, Panama City, FL 32407 USA (e-mail: vatana.an@navy.mil; frank.crosby@navy.mil).

Z. Qu is with the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816 USA (e-mail: qu@ucf.edu).

R. Roberts is with the Department of Electrical and Computer Engineering, Florida A&M University–Florida State University, Tallahassee, FL 32310-6046 USA (e-mail: rroberts@eng.fsu.edu).

V. An is with the Department of Computer Science, University of West Florida, Pensacola, FL 32514 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2018.2806840

TABLE I
COMMON ABBREVIATIONS

Abbreviations	Long form
1. ACD	Adaptive Circle Diameter
2. ACC	Adaptive Curvature Control
3. BPP	Baseline Path Planning
4. CP	Coverage Path
5. CPL	Coverage Path Length
6. DC	Discontinuous to Continuous
7. DLL	Doubly Link List
8. DT	Delaunay Triangulation
9. EP	Extreme Point
10. HLT	Horizontal Line Test
11. HV	Hole Vertex
12. LOP	Local Observer Planning
13. LS	Line Segment
14. LS(A,B)	Line Segment With Endpoints A and B
15. LP(A,B)	Line Passing Through Endpoints A and B
16. NN	Nearest Neighbor
17. OP	Observer Placement
18. OW	Observer Waypoint
19. PNWCC	Previous Next Waypoint Coverage Constraint
20. POR	Path and Observer Replanning
21. RBOP	Row Based Observer Placement
22. RG	Region
23. RT	Regular Triangulation
24. SCA	Static Collision Avoidance
25. SMS	Sensor Maximum Square
26. SMSD	Sensor Maximum Square Dimension
27. SPR	Smooth Path Replanning
28. TBCPP	Triangulation-Based Coverage Path Planning
29. TR	Target Region
30. TSP	Traveling Salesman Problem
31. VLS	Visibility Line Segment
32. VLT	Vertical Line Test
33. VP	Visible Polygon
34. VPV	Visible Polygon Vertex
35. WP	Waypoint

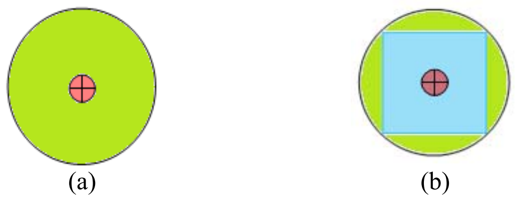


Fig. 1. Robot and its circular sensor. (a) Robot enveloped by the sensor in green. (b) Maximum square enveloped by a circle.

number. For example, an ith object with radius R_i will be represented by obstacles centered at point O_i.

Assumption 4: The set Ω to be covered is two-dimensionally connected, with respect to a disk of the robot's radius R_r. For example, consider the target region (TR) in Fig. 2 with nine disks.

A. Patrolling Control Problem

Given a TR like Fig. 2 with a finite number of static objects, how do we design a differentiable and continuous path for a nonholonomic mobile sensor with range, coverage, movement, and time constraints that can sweep the given area without collision? Given an initial position and orientation of the robot represented by P_i and θ_i and the environment under Assumptions 1–4, we find a smooth CP which the robot moves collision-free, and covers all points in the set Ω over time. Mathematically, the problem is to determine a differentiable path $s(t)$ by ensuring conditions represented by (1) and (2) hold

$$\min_{t\in[t_0,t_0+T]} \|q - s(t)\|\phi(q,t) \le R_r \quad \forall q \in \Omega \tag{1}$$

where $\phi(q,t)$ is a weighting function which can be chosen at design time. q is the point in the set. T is the time for the robot to complete its maneuver between adjacent pair of points

$$\|s(t) - O_i\|_{t\in[t_0,t_0+T]} \ge R_r + R_i \;\forall i \in \{1,\ldots,n\}. \tag{2}$$

The patrolling control problem is solved with the four phases of the triangulation-based CP planning (TBCPP) approach as shown in Table II which consists of a set of existing and newly developed algorithms. More detail descriptions of the newly developed algorithms are tabulated in Table III. Table IV lists the algorithms used in each phase.

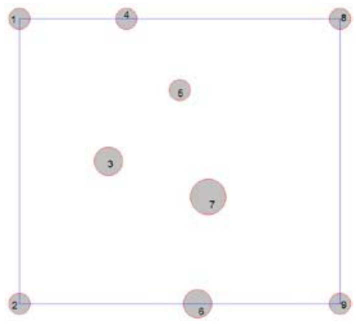


Fig. 2. Sample TR to be covered with finite number of disks.

TABLE II
FOUR PHASES OF THE TBCPP APPROACH AND THEIR DESCRIPTIONS

1. **Local Observer Planning (LOP)** – Developed to place observer(s) in each disjoint sub region of the TR which is called regular triangulation (RT). Two different algorithms are developed to place observer in the RT depending on the relative size of the RT and R_s.
2. **Baseline Path Planning (BPP)** – Developed to generate a CP based on the observers found in phase 1. The CP obtained in this phase may have a collision with the known disks in the TR. All collisions may be removed as required while completely observing the TR. If collisions are not removed in this phase, they will have to be removed in the next phase.
3. **Path and Observers Replanning (POR)** – Developed to reduce the CP length while also reducing the number of observer required to observe the TR. All collisions will be removed in this phase if they are not already removed in phase 2. This phase is optional for efficiency. A Doubly Linked List data structure is used to systematically organize and access data.
4. **Smooth Path Replanning (SPR)** – Developed to smooth the coverage path for continuity and efficiency while still maintaining complete coverage of the TR.

III. PRELIMINARY

In this section, we present the organization of the TBCPP approach and some of the existing algorithms used. For organizational simplicity, we will be presenting the algorithms in the logical order as they appear in Table II. The six novel algorithms contributed in this paper, as tabulated in Table III, are used in different phases of the overall TBCPP approach.

TABLE III
AUXILIARY ALGORITHMS USED BY THE TBCPP ALGORITHM

1. Visible Polygon (VP) – developed to compute the convex polygon inside the RT from which a sufficient range observer can be placed within the convex polygon to observe the entire RT.
2. Row Based Observers Placement (RBOP) – developed to find a set of observers within the RT that can collectively observe the entire RT. This algorithm is used when an RT is relatively larger than the sensing range, Rs.
3. Static Collision Avoidance (SCA) – developed to replace the colliding line segment of the CP with two new line segments that detour around the disk that involve in collision.
4. Previous Next Waypoint Coverage Constraint (PNWCC) – developed to reduce the path length while maintaining complete coverage. This algorithm may move observers within the TR or delete them as needed.
5. Adaptive Circle (AC) – developed to find smooth curvature between three waypoints and the two LSs connecting them.
6. Discontinous to Continuous (DC) – developed to transform the linear spline CP, piecewise continuous CP, into smooth CP while maintaining the same observer location.

TABLE IV
FOUR PHASES OF TBCPP APPROACH AND THEIR SUB ALGORITHMS

1. Local Observer Planning
(E) Delaunay Triangulation
(D) Visible Polygon
(D) Row Based Observer Placement
2. Baseline Path Planning
(E) Nearest Neighbor
(D) Static Collision Avoidance
3. Path and Observers Replanning
(D) Previous Next Waypoint Coverage Constraint
(D) Static Collision Avoidance
4. Smooth Path Replanning
(D) Adaptive Circle algorithm
(D) Discontinuous to Continuous algorithm

In Table IV, the subalgorithms under each of the four phases of the TBCPP approach are designated with (E) for existing algorithms and with (D) as the new algorithms developed for this paper which represent our new contributions. The following sections discuss the existing algorithms, the Big-O notation, and the doubly linked list (DLL).

A. Delaunay Triangulation Algorithm

Delaunay triangulation (DT) algorithm is an existing algorithm employed to partition the TR into several disjoined regular triangulations (RTs). This step can be completed in $N\log(N)$ time [15]. The sample TR in Fig. 2 with nine disks can be partitioned into ten RTs as shown in Fig. 3. Partitioning for N number of vertices obey Euler's formula. This is also true for N number of disks. While Euler's formula equates the relationship of the number of faces, vertices, and edges, they are related here in terms of the number of RTs, disks, and edges, respectively. Note that edges generated by DT are actually line segments (LSs) in this paper because all of them are straight. For clarity, edge in (3a) will be referred to as LS. All other reference of edge that is not straight from here on forth, will be referred to as curve segment (CS)

$$F + V - E = 1 \tag{3a}$$

$$\mathrm{RT} + \mathrm{DISK} - \mathrm{LS} = 1. \tag{3b}$$

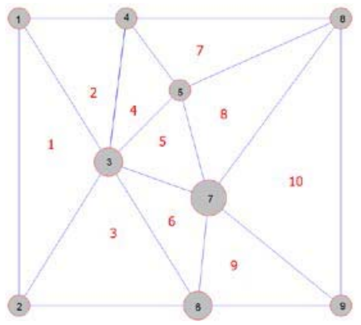


Fig. 3. RTs of the sample TR.

Fig. 3 illustrates several different types of RTs of the sample TR. As a side note, Fig. 3 has 9 disks, 10 RTs, and 18 LSs. This obeys (3) as formulated by Euler. An RT is very similar to a triangle, except that an RT is formed by three disks of varying size while a triangle is formed by three points. An RT is nonconvex while a triangle is convex. DT algorithm is used by the local observer planning (LOP) phase of the TBCPP approach.

B. Nearest Neighbor Algorithm

A number of Traveling Salesman algorithms are published in the literature. Different variant of Traveling Salesman algorithms may generate different path length and may encounter different level of complexity. The best heuristic algorithm that guarantees a path length close to the optimal solution, but with complicate running time is the Sanjeev Arora algorithm. To simplify the problem, the CP in this paper is connected with the nearest neighbor (NN) variant of the Traveling Salesman Problem (TSP). The running time for the NN algorithm is $O(n^2)$ [16].

C. Big-O Notation

In 1892, German mathematician introduced the big-O notation to compute the complexity of algorithms. The notation $O(N)$ represents the linear running time, also known as linear complexity, of the algorithm. The big-O notation is intended to express the qualitative behavior of the algorithm, instead of the quantitative behavior [17]. In term of qualitative analysis, the values of $O(N)$, $O(2N)$, and $2(O(N))$ are consider the same. The expression $O(K(N^2 + N))$, with K being a constant, is qualitatively similar to $O(N^2)$ and generally express with the later notation [17], [18].

D. Doubly Linked List

The previous next waypoint coverage constraint (PNWCC) algorithm in the third phase of the TBCPP approach requires a data structure to accommodate all size of sensing radii that observe all size of RT with the ability to delete or modify observer's location as necessary. The PNWCC algorithm requires a sophisticated data structure to keep track of all transaction. A DLL is a perfect choice for our novel PNWCC tour replanning algorithm due to fast element access, quick modification, addition, or deletion of observers, fast update time for tail observer or lead observer, and quick traversal time. A representation of a subset of the tour in a DLL will be discussed in more detail in Section IV-C. Insertion or deletion at either head observer or tail observer is done in constant time. Observer data structure access is done in linear time [17].

IV. Triangulation-Based Coverage Path Planning Approach

The TBCPP approach consists of four phases as seen in Table II. The overall running time of the TBCPP approach is quadratic which is considered fast. The slowest activity in the TBCPP approach is the linking of all waypoints (WPs) with the TSP's NN algorithm. This activity can be improved with better algorithms already exist in the literature.

A. Local Observer Planning

LOP is the first phase of the TBCPP approach and it consists of three subalgorithms: 1) DT; 2) visible polygon (VP); and 3) row-based observer placement (RBOP). The VP algorithm allows the planner to check if the sensing range available is sufficiently large to observe the RT. If the sensing range is sufficiently large, then only one observer is required for an RT. Else, the planner has to execute the RBOP algorithm to place sufficient number of observers for an RT. This process is repeated until all RTs are observed.

1) Visible Polygon Algorithm: A number of definitions have to be defined to discuss the VP algorithm. They are the extreme point (EP), the visibility LS (VLS), the VP, and the VP's vertex (VPV). Fig. 4(a) shows RT 5 which is formed by disks 3, 5, and 7 and line segments LS(3,5), LS(3,7), and LS(5,7). LS(3,5) have two EPs spawn by disks 3 and 5. All EPs of the RT are computed by (4) and (5). The VP of the RT is computed as a function of the 6 EPs within the RT and the LSs forming the RT. Fig. 4 illustrated the process of computing a VP of the RT. RT 5 is a typical RT with six EPs form by the three disks and three LSs. The six EPs are shown in red dots in Fig. 4(b) and they are labeled as points A, B, C, D, E, and F. Computing the VP is easier to visualize by transforming the RT into a triangle. This is done by reducing all disks of the RT into points as shown in Fig. 4(c). An enclosing rectangle is introduced in Fig. 4(d) to simplify the computation of the VP and to determine the length of the VLS required.

The rectangle enclosing the triangle in Fig. 4(d) can be determined with the min and max functions to find the minimum and maximum coordinates in the x-axis and the y-axis of the selected triangle's coordinates. The coordinates of the enclosing rectangle are $(x_{\min}, y_{\min})$, $(x_{\min}, y_{\max})$, $(x_{\max}, y_{\min})$, and $(x_{\max}, y_{\max})$. The VLS within an RT is found from the EP and the LS that the EP is on.

Fig. 4(g) shows the VLS in green color which is spawn by EP A and LS(3,5). VLS A is perpendicular to LS(3,5) at EP A. Every VLS is perpendicular to the LS that spawn them at the EP being considered. VLS B is perpendicular to LS(3,7) because EP B is on LS(3,7). Fig. 4(h) shows the new VP in yellow and the subarea to be removed due to EP A in red. Every VLS due to an EP only make the initial VP getting smaller and smaller. Since every LS of the RT has two EPs, the VP can be found from the relative positions of the two EPs of every LS. For example, the initial VP due to EP A is above the VLS A because the other EP on the LS that EP A is on is EP C which is above EP A. The overall VP of the RT has to be determined from all six EPs and it would be smaller than the triangle of all three disks when the three disks are reduced to points.

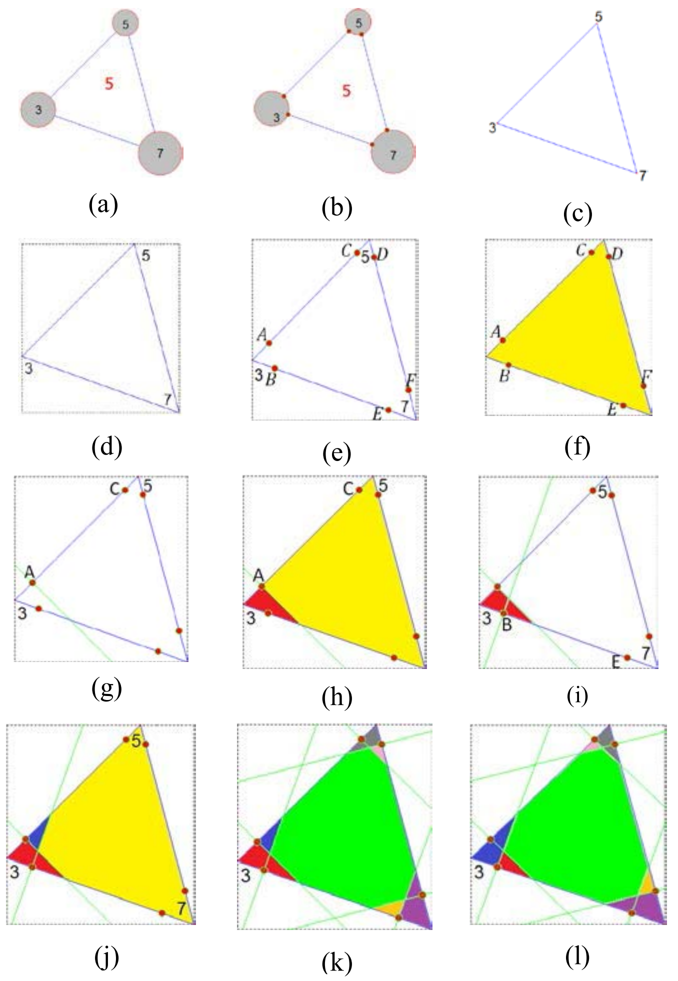


Fig. 4. Transition from RT to VP from left to right and from up down to bottom. (a) RT 5. (b) RT 5 and its 6 EPs. (c) Triangle 5 (T5). (d) Rectangle enclosing T5. (e) T5 and 6 EPs. (f) Initial VP in yellow. (g) VLS due to EP A. (h) VP due to EP A. (i) VLSs due to A and B. (j) VP due to A and B. (k) VP due to all 6 EPs. (l) VP due to all 6 EPs.

Fig. 4(f) illustrates the initial VP before considering any EP's effect. At this point, the number of VPVs is 3 which is the same as the triangle's vertices. Fig. 4(i) shows the VLS due to EP B which generate a subregion shown in blue to be removed from the initial VP as shown in Fig. 4(j). The subregions to be removed due to EP A and EP B would be different in term of size and shape if the EP B were to generate the VLS before EP A. Fig. 4(k) illustrates the case with EP A being processed before EP B. Fig. 4(l) illustrates the case with EP B being processed before EP A.

Once all EPs are processed, the resulting VP is the same regardless which EP is processed first. The green convex polygon in Fig. 4(k) and (l) are the resulting VP after all 6 EPs are considered and it has 9 VPVs.

TABLE V
RT'S VP ALGORITHM

```
Input:  An RT of disks i, j, and k (e.g. RT 5 in Fig. 4(a)).
Output:  The convex visible polygon.
Algorithm Steps Description:
1.   Compute all six EPs of the RT.
2.   Reduce all disks of the RT into points to get a triangle.
3.   Initialize the VP to the triangle in step 2.
4.   For each of the six EPs found in step 1, compute the corresponding VLS.
     Compute the intersection of the VLS with the VP. Remove the region to
     the left, right, below, or above the two EPs on the LS that spawn the
     VLS. Update the VP.
```

As shown in Table V, four steps are required to compute the VP. Step 1 of the VP algorithm can be computed from the disk equation and the line equation in (4) and (5), respectively. The results for RT 5 are EPs A, B, C, D, and E. x_{ic}, y_{ic}, and R_i represent the coordinate of the disk i and its radius, respectively.

For this particular RT, all points resulting from VLSs intersecting with each other and with the RT's LSs are the VPVs. For our application of constraining the VP to be within the RT, the maximum number of vertices that formed the VP is 9. The minimum number of vertices is 3. Fig. 6 will show the VP of every RT for the given TR. An existence of a single VP within an RT guaranteed a single observer with sufficient range to observe the whole RT as long as the observer is in the convex VP. Fig. 6(a) illustrates all RTs and their VPs. The VP algorithm allows the computation of the visible region and nonvisible region of the RT to be determined. For example, if an observer is placed inside the nonvisible region (region inside of the RT, but outside of the VP) then the entire RT cannot be observed, even when the sensing range is exceeding the constraint in (8)

$$(x - x_{ic})^2 + (y - y_{ic})^2 = R_i^2 \tag{4}$$

$$y - y_{ic} = m(x - x_{ic}) + b. \tag{5}$$

In local planning of a single RT, without knowledge of the entire environment, it is a good idea to compute the centroid of the VP as the position of the observer. The position of the observer can be moved to optimize curvature or distance requirement through our novel PNWCC algorithm which is employed in phase 3.

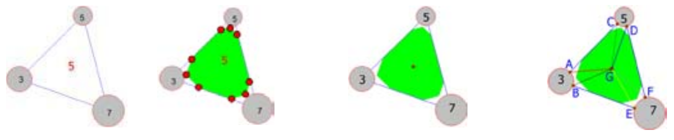
The centroid of a VP can be computed with (6), where A_{VP}, X_C, Y_C, and n represent an area, the centroid's coordinate, and the number of VPVs for the VP, respectively [19]. (x_i, y_i) are representing the coordinates of the VPV for all values of i from 0 to $n-1$

$$A_{\text{VP}} = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i) \tag{6a}$$

$$X_C = \frac{1}{6A_{\text{VP}}}\sum_{i=0}^{n-1}(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \tag{6b}$$

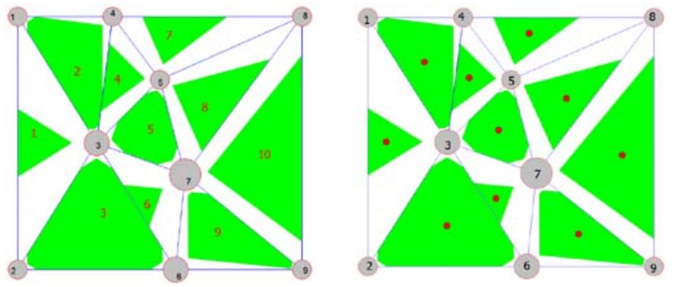
$$Y_C = \frac{1}{6A_{\text{VP}}}\sum_{i=0}^{n-1}(y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i). \tag{6c}$$

Fig. 5(d) shows that in Fig. 5(d) shows that in order for the RT to be observable by a single observer, R_s must be larger than or equal to the maximum length expressed in (8a), where e_i represent the EPs A, B, C, D, E, and F. G represents the centroid of the VP. Computing Rs in (8a) only required six quantities. Equation (7) represents the Euclidean length between any two points. Equation (8b) provides the condition that the sensor can be anywhere in the VP and able to observe the RT. v_j is representing the VPV. For Fig 5(b), there are nine VPVs in red. Rs in (8b) is larger than Rs in (8a). Computing the minimum Rs for observer to be anywhere in the RT require Rs to meet the requirement in (8b) as well as computing 54 quantities for verification because of six EPs and nine VPVs, the maximum number of VPVs per RT. RT 1 has only three VPVs and three is the minimum number of VPV for a VP



(a) (b) (c) (d)

Fig. 5. Typical RT, its VP, its VP's centroid, and length. (a) RT 5. (b) RT 5's VP. (c) VP 5's centroid. (d) L(EP,G).



(a) (b)

Fig. 6. RTs, VPs, and the VP's centroid in the TR. (a) TR's RTs and their VPs. (b) VPs' centroids.

$$L(P_1, P_2) = \sqrt{\left(P_{1,x} - P_{2,x}\right)^2 + \left(P_{1,y} - P_{2,y}\right)^2} \tag{7}$$

$$R_s \geq \max_{e_i \in E} L(e_i, G) \tag{8a}$$

$$R_s \geq \max_{e_i \in E, v_j \in V} L\left(e_i, v_j\right). \tag{8b}$$

2) Row-Based Observers Placement Algorithm: Multiple observers in an RT may be required for complete coverage. In this section, sufficient conditions to observe an RT is formulated, derived, and proved. The RBOP algorithm computes necessary quantities to find the sufficient number of observers to observe the RT.

3) Row-Based Observers Placement Theorem: Given any RT of three disks with one VP, as shown in Fig. 7, a sufficient number of circular observers needed for complete coverage is $\sum_{n=1}^{\text{row}_\Delta} \text{row}_n$, where row_n is the number of circular observers in row n. row_Δ is the total number of row in the RT.

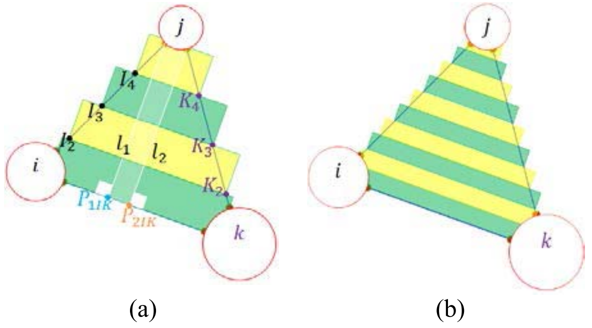


Fig. 7. Row-based approach to find sufficient number of observers. (a) Four rows of observers. (b) Ten rows of observers.

TABLE VI
ROW-BASED OBSERVERS PLACEMENT ALGORITHM

Input: An RT of disks i, j, and k, and an observer's sensing range, R_s. The centers of disks i, j, and k are points I, J, and K respectively.
Output: All observer points and their coordinates and their region to be observed.
Algorithm Steps Description:

1. Select $LS(I,K)$ as the base of the RT. Once $LS(I,K)$ is selected as the base of the RT, then the other two LSs are indirectly selected as the height of the RT.
2. Compute l_1 and l_2 such that they are perpendicular to the base of the RT at points P_{1IK} and P_{2IK}.
3. Compute row_Δ, and then compute $I_1, I_2, \dots, I_{row_\Delta}$. Also compute $K_1, K_2, \dots, K_{row_\Delta}$.
4. Compute $row_1, row_2, \dots, row_{max}.max = row_\Delta$.
5. Compute the observer's rectangle in each row. The centroid of each of the rectangle is the observer's location. Note that the sides of the rectangle are equal to the SMSD or smaller.

Proof: It is easy to observe the LSs' relation in Fig. 7(a) that $l_j \perp l_{IK}$ for $j = 1, 2$ and $l_1 \parallel l_2$. l_{IK} is the line from centers of disk i to disk k

$$\text{row}_\Delta = \text{INT}\left[\frac{\max(L(l_1), L(l_2))}{\sqrt{2}R_s}\right] \tag{9}$$

$$\text{row}_n = \text{INT}\left[\frac{\max\big(L(I_n, K_n), L\big(I_{(n+1)}, K_{(n+1)}\big)\big)}{\sqrt{2}R_s}\right]. \tag{10}$$

I_n and K_n are points that can be found using the disk's and the line's formulas. Lines l_{IK}, l_{IJ}, l_{JK}, and l_j can be computed since the origin of disks i, j, and k are known. $L(P_1, P_2)$ is the Euclidean distance between points P_1 and P_2 as expressed in (7). Note that $L(l_1) = L(C, P_{1IK})$ and $L(l_2) = L(D, P_{2IK})$. C and D are the EPs of the RT as shown in Fig. 5(d). Note that the largest rectangle that a circular sensor can envelop is a square as shown in Fig. 1(b). ■

With the RBOP algorithm in Table VI, RT 5 as input results in 11 observers in Fig. 8(a). Each of the rectangles covering the cell area is smaller than the SMS [see Fig. 1(b)].

As a result, the number of observer in each row as express in (10) usually result in smaller area being observe than the SMS which is a rectangle with one of the sides equal to the side of the SMS dimension (SMSD) and the other side is usually smaller. With the configuration of disks i, j, and k in Fig. 8 which correspond to disks 3, 5, and 7 as tabulated in Table XII, $l_1 = 230.67$ and $l_2 = 228.04$. If $R_s = 50$, then the SMSD is 70.71. The total number of row of observer is then

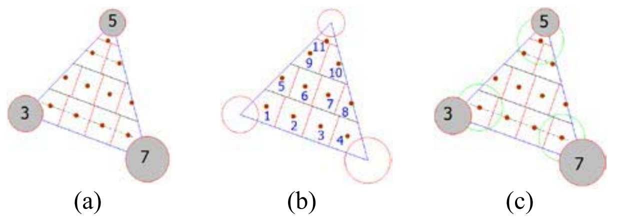


Fig. 8. RBOP in RT 5 with Rs = 50. Observers are ordered from left to right and from bottom to up. (a) RBOP in RT 5. (b) RT 5's observers. (c) Three sensors on.

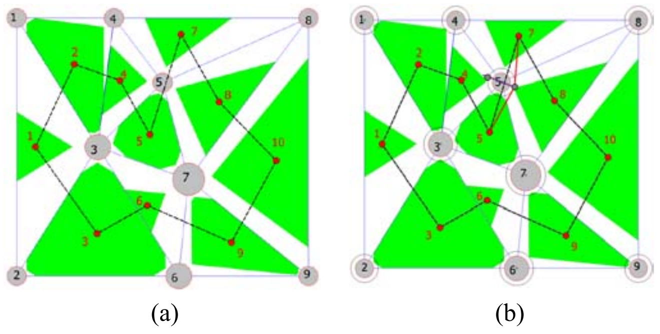


Fig. 9. TR's CP due to TSP's NN algorithm. (a) TR's centroidal CP. (b) CP with collision correction.

$\text{row}_\Delta = 4$. The number of observers in row 1 is $\text{row}_1 = 4$. Likewise, the number of observers in row 2 is $\text{row}_2 = 4$. Rows 3 and 4 only require two observers and one observer, respectively. Note that observing region in each row may be of different size or shape due to the configuration of the row in an RT.

B. Baseline Path Planning

Once the TR with known number of disks has been partitioned, depending on the radius of the sensing range, a sufficient number of observers can be computed to observe the TR. For sufficient sensing range example shown in Fig. 6(b), only one observer is required per RT. Each and every observer within the RT can be connected to form a baseline CP. Many algorithms may be implemented to connect all observer points (OPs). Since a TSP's NN algorithm is implemented, many potential CP is obtained. The shortest CP may be selected without consideration of collision as it can be fixed later. Equation (11) is the TSP's equation that determines the length of the CP with m number of WPs. At this point, all WPs are OPs. Fig. 9(a) illustrates the CP resulting from connecting all OPs in Fig. 6(b). The CP in clockwise direction is WPs 1, 2, 4, 5, 7, 8, 10, 9, 6, 3, and 1. Since the algorithm is constructing a tour, the trip starts and ends at the same point

$$s(x) = L\big(p_{\pi(m)}, p_{\pi(1)}\big) + \sum_{i=1}^{m-1} L\big(p_{\pi(i)}, p_{\pi(i+1)}\big). \tag{11}$$

Every LS of the CP has to be tested for collision with the obstacle or disk. It is obvious that LS(5,7) is colliding with disk 5. A simple method to avoid collision with any known static disk is the static collision avoidance (SCA) algorithm.

TABLE VII
SCA ALGORITHM

Input: A line segment with two end points I, J and the disk involved in collision, disk k.
Output: Two new LSs that detour the disk involve in collision, $LS(I,K)$ and $LS(K,J)$.
Algorithm Steps Description:
1. Compute $LS(K_B,K_G)$ that pass through the center of disk k and is perpendiculartoLS(I,J). Note that K_B and K_G are on the enlarged circle of disk k and the robot's radius.$K_B \le LS(I,J) < K_G$.
2. Compare and select the point K_B or $K_G$that result in shorter distance, $LS(I,K_B)$ and $LS(K_B,J)$or $LS(I,K_G)$ and $LS(K_G,J)$. Name the selected point as K.

1) Static Collision Avoidance Algorithm: The SCA algorithm finds an equation of the line that is perpendicular to the LS that involve in collision and that also passes through the center of the disk [see Fig. 9(b)]. Any collision involved with a disk will have two possible solutions. The goal is to pick the intermediate point on the disk that provides the shorter distance. Fig. 9(b) illustrates two points in gray dots that provide the solution and they are on the enlarged disk 5. For example, collision in Fig. 9(b), the point with the shorter distance is to the right of the original LS that encounter collision. Note that all disks in Fig. 9(b) are enlarged by the size of the robot. Fig. 9(a) illustrated the robot as a point, no disk enlargement is required. As illustrated with Fig. 9's example, a collision correction required an insertion of a WP into the CP. This means that two LSs are replacing a colliding LS by inserting a new WP to detour the collision. This new WP is not an observer waypoint (OW). Additional details of the SCA algorithm can be found in Table VII.

C. Path and Observers Replanning

In a PNWCC algorithm, the current OP also known as OW or simply as WP is consider to improve the CP while the previous and the next WPs are used as a constraint to measure improvement and to manage positional change of the current WP. Improvements of the CP include reducing the distance as well as the angle. Each and every WP on the CP may move to a different position or delete.

As a result, the number of OPs may also be reduced which mean energy saving, computational reduction, and time efficient. There are two variant of PNWCC algorithms: RBOP PNWCC and VP PNWCC, also referred to as PNWCC Algorithms 1 and 2. Fig. 10 illustrated the RBOP PNWCC algorithm which is tabulated in Table VIII. In Fig. 10(b), four hole vertices (HVs) surface as a result of turning on WPs 5 and 7 and turning off WP 6. The goal is to modify the location of WP 6 to minimize path length while maintaining complete coverage of the regions that were originally covered by the three WPs. Fig. 11(c) shows that the new OW covers all HVs.

The three regions are represented by regions 5, 6, and 7 in Fig. 8(b). WP 8 in region 8 is irrelevant at this point because it is not part of the previous–current–next WPs as presented in the CP in Fig. 10(a). WPs 9, 10, 1, 2, and 3 are also not considered at this point because of the complexity of keeping up with all HVs.

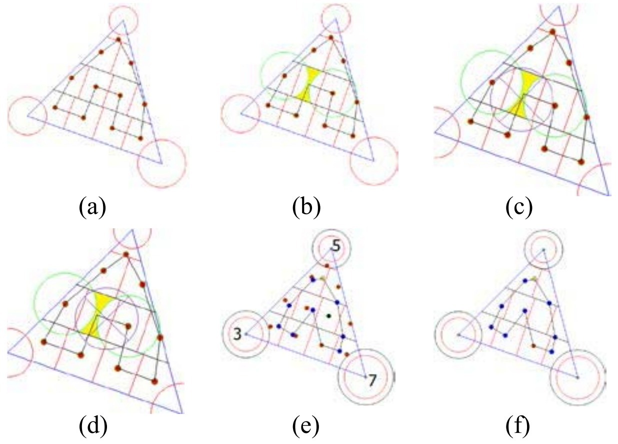


Fig. 10. Coverage of an RT prior to and post PNWCC algorithm. (a) Selected CP with TSP's NN algorithm. (b) Row 2 coverage with OWs 5 and 7 turned on. (c) Potential placement of OW 5. (d) Potential placement of OW 5. (e) Old and new observers in the RT. (f) Modified CP due to new observers.

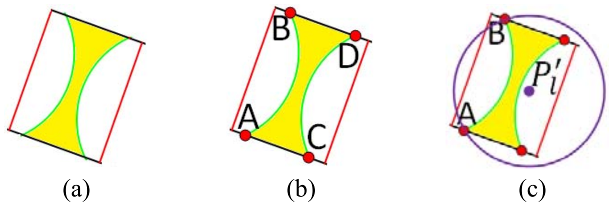


Fig. 11. WP 6's hole and the new WP 6 to cover the hole. (a) Hole 6. (b) Hole 6's vertices. (c) New OW.

Fig. 10(c) and (d) presented two potential locations of the modified WP 6 when only WPs 5, 6, and 7 are considered. Fig. 10(d) shows the new location of WP 6, also shown in Fig. 11(c), because it results in the shortest distance among all possibilities and constraints due to the hole. This distance improvement is formulated in (12). Once the new WP 6 is found, the algorithm continues to select the next WP in the CP's queue until it ends at the very first WP that it begin with. That is, when the algorithm stops running until the next iteration. The next WP in the CP's queue after WP 6 is WP 2 according to the selected CP which is shown in Fig. 10(a) and the convention that the algorithm executes in clockwise ordering. For example, clockwise ordering of the CP in Fig. 10(a) is WP 6, 2, 1, 5, 9, 11, 10, 8, 4, 3, 7, and 6. In this case, the CP may also be equivalently referred to as tour.

The PNWCC algorithm moved WP 6 slightly to the right to improve the path length from WP 7 to WP 6 and then to WP 2. The *k*, *l*, and *m* in (12) correspond to WPs 7, 6, and 2, respectively, in this case. While many WPs were moved to a different position to improve the path length and the sharpness of the path, the last WP in the CP was deleted as it is no longer necessary. All of this is done in a single iteration of the PNWCC algorithm. The PNWCC algorithm may be running in several iterations to get the best result. Fig. 10(e) shows the new WP, WP modified by the PNWCC algorithm, in blue and the original WP, WP in the baseline CP in red. The yellow WP is the new WP 11. Note that the original WP 11 is colliding

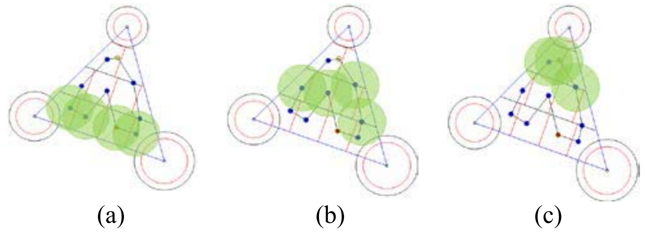


Fig. 12. Coverage of an RT after one iteration of PNWCC algorithm. (a) Observed row 1. (b) Observed row 2. (c) Observed rows 3 and 4.

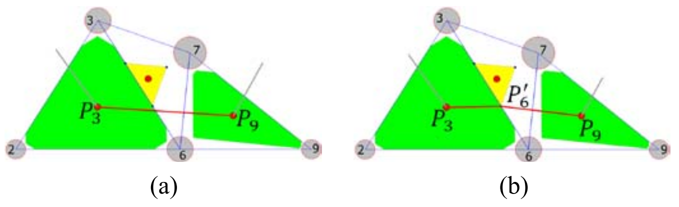


Fig. 13. Example of VPV maintaining visibility. (a) Straightline is not observable. (b) Shortest observable path.

with disk 5 if the robot is not a point and has a radius larger than 5 units. When the observers were found with the RBOP algorithm, it is assumed that the sensor or the robot is a point and in this case collision did not happen

$$L(P_k, P_l, P_m) = L(P_k, P_l) + L(P_l, P_m) \tag{12a}$$

$$L(P_k, P_l, P_m) = L(P_k, P_m) \text{ if } P_l \text{ is collinear with } P_k, P_m \tag{12b}$$

$$L\big(P_k, P'_l\big) + L\big(P'_l, P_m\big) < L(P_k, P_l) + L(P_l, P_m). \tag{12c}$$

One complete iteration of the PNWCC algorithm shows drastic improvement as compare with Fig. 10(a) and (f). Fig. 12 also shows that all rows are still observable with the set of new WPs. If the sensing range Rs is equal to or greater than the constraint in (8b), then a straight line test may be checked to see if (12a), LS(P_k, P_m), intersect with the current VP as in Fig. 13(a), $k = 9$ and $m = 3$ and Fig. 14(a), $k = 8$ and $m = 9$. If no intersection occurs, as is the case in Fig. 13(a), then the minimum distance solution for this previous–current–next WPs does not exist. Alternate solutions may be found with the vertices of the VP as is the case in Fig. 13(b). Originally, the algorithm does not know which of the three VP's vertices is shortest. All three vertices will be compared and the shortest is considered.

The example in Fig. 10 is found in step 4 of the PNWCC Algorithm 1. The examples in Figs. 13 and 14 are found in step 5 of the PNWCC Algorithm 2. Note that PNWCC for RBOP and VP have slightly different inputs. After running 1 iteration of the PNWCC Algorithm 1, the number of WP may be reduced from m to m', $m' < m$. As seen in Fig. 10(a) and (f), the length $s(x)$ is reduced to $s'(x)$. Situation like Fig. 10 provides optimization of coverage path length (CPL) and curvature as well as reduction of the number of observers. The CP in Fig. 15 is the improved version of the CP in Fig. 9 due to PNWCC algorithm 2. Because WP5 and WP7 do not change position, the LS(5,7) still collide with disk 5 the same way it did prior to PNWCC algorithm. Fig. 15(b) shows the version without collision due to the SCA algorithm. The CP's clockwise order in Fig. 15(b) is WPs 1, 2, 4, 5, 11, 7, 8, 10, 9, 6, and 1

$$s'(x) = L\big(p_{\pi(m')}, p_{\pi(1)}\big) + \sum_{i=1}^{m'-1} L\big(p_{\pi(i)}, p_{\pi(i+1)}\big). \tag{13}$$

The colliding link is break into two LSs. Because there were 10 WPs before considering the collision, the WP to avoid the collision is named number 11 since it is the next number after 10. Of the 11 WPs, 10 WPs are OW while 1 WP is not an OW. WP 11 is a collision avoidance WP. In Fig. 15, the original OWs are colored gray to denote deletion from the CP. Gray and yellow WPs are not OWs. Blue OWs are new WP. Red OWs are unchanged.

TABLE VIII
PREVIOUS–NEXT WPS COVERAGE CONSTRAINT ALGORITHM 1

Input: A linear spline tour, $s(x) = G(V, E)$, consisting of WPs with known beginning and ending, (e.g. CP in Fig. 10(a)), and all regions in all RTs.
Output: A better linear spline tour, $G'(V', E')$, with fewer WPs and with shorter tour length (e.g. CP in Fig. 10(f)).
Algorithm Steps Description:

1. Initialize all Regions in the RT to COVERED state, number of HVs for all WPs to zero, and all WP's state to MODIFIABLE.
2. Select a WP to begin with if it state is MODIFIABLE.
3. If the select WP is MODIFIABLE, remove the selected WP from the Region it is associated with and then initialize the Region it is supposed to be covered to NOT COVER. Else, go to step 7.
4. Identify the row that the select WP is in. If more than three WPs are in the row, turn on the two closest neighboring WPs while turn off the selected WP (see Fig.10(b)-(d)). Identify all HVs that surface and compute their cartesian coordinates, see Fig. 10 and 11 for illustration. Else if only one WP is in the row, turn off the selected WP. Identify all HVs that surface and compute their cartesian coordinates.
5. Since the sensor is circular, only two HVs found in the previous step may be used at a time to compute the new WP. Use equations (12c) to find the best new WP P_l' to replace P_l. This may require 2 to the power of the number of HVs. The new WP must be able to restore the Region it has altered to COVERED state by verifying that all HVs are in the circle of the new WP as well as shorten the distance in equation (12a). If the new WP cannot make improvement, keep the old WP. The new WP may be in the same Region or move to another Region. Once the new WP is determined, update the DLL of the new WP to NOT MODIFIABLE and their HV counter to 0.
6. If the next WP in the tour is not the last WP, select the next WP in the tour and repeat steps 3, 4, and 5. Else go to step 7.
7. All WPs have been considered for improvement. Check each and every Region for excess number of WP(s) and remove them. The new tour $G'(V', E')$ is a set of piecewise continuous functions, and it is at least as good as the old tour or better. See Fig. 10(f).

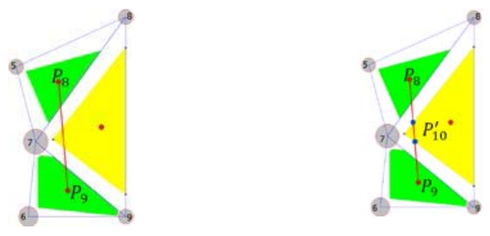


Fig. 14. Example of straight line maintaining visibility. (a) Straight line, $\overline{P_8P_9}$, is observable. (b) Observable points between two blue dots inclusive.

TABLE IX
PREVIOUS–NEXT WPs COVERAGE CONSTRAINT ALGORITHM 2

Input: A linear spline tour, $s(x) = G(V, E)$, consisting of WPs with known beginning and ending, and all RTs and their respective VPs.
Output: A better linear spline tour $G'(V', E')$, with fewer WPs and with shorter tour length.
Algorithm Steps Description:

1. Initialize all RTs in the TR to COVERED state, number of HVs for all WPs to zero, and all WP's state to MODIFIABLE.
2. Select a WP to begin with if it state is MODIFIABLE.
3. If the select WP is MODIFIABLE, remove the selected WP from the RT it is associated with and then initialize the RT it is supposed to be covered to NOT COVER. Else, go to step 8.
4. Check the number of WPs in the RT. If the number of WP is less than 4 and R_s is less than R_s in (8a), then turn on all of the WPs in the RT while turn off the selected WP. Identify all HVs that surface as a result of removing the WP in this step and compute their Cartesian coordinates.
5. If an RT has only 1 WP and R_s meet the criteria in (8b), compute the intersection of straightline $LS(P_k, P_m)$ and the VP. If no intersection found, find the new WP with the vertices of the VP and equation (12c). Keep WP P_l if neither the intersection nor the vertices condition is resulting in shorter $LS(P_k, P_m)$. Go to step 7.
6. Since the sensor is circular, only two HVs found in the previous step may be used at a time to compute the new WP P_l'. Use equations (12c) to find the best new WP to replace the removed WP. This may require 2 to the power of the number of HVs. The new WP must be able to restore the RT(s) it has altered to COVERED state as well as shorten the distance of the WP it has modified. If the new WP cannot make improvement, keep the old WP. The new WP may be in the same RT or move to another RT. Once the new WP is determined, update the DLL of the new WP to NOT MODIFIABLE and their HV counter to 0.
7. If the next WP in the tour is not the last WP, select the next WP in the tour and repeat steps 3, 4, 5, and 6. Else go to step 8.
8. All WPs have been considered for improvement. Check each and every RT for excess number of WP(s) and remove them. The new tour $G'(V', E')$ is a set of piecewise continuous functions, and it is at least as good as the old tour or better.

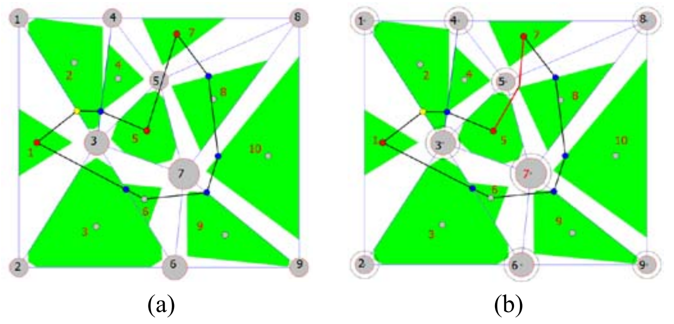


Fig. 15. One iteration of tour reduction, PNWCC algorithm. (a) Colliding PNWCC CP. (b) Collision-free PNWCC CP.

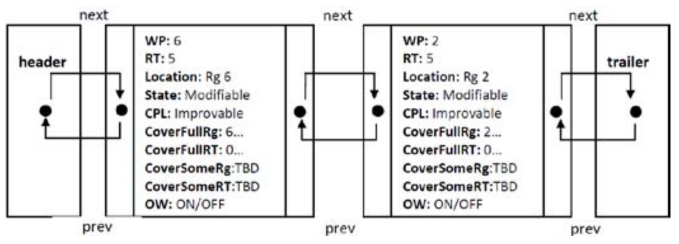


Fig. 16. DLL for PNWCC algorithm.

The DLL presents in Fig. 16 illustrate some control elements of the WP's data structures. CoverFullRg and CoverFullRT represent full coverage of the region and an RT, respectively. CoverSomeRg and CoverSomeRT represent some coverage of the region and an RT, respectively. These optional parameters are not covered in this paper. They are present to illustrate the capability of DLL data structure. State and CPL are Boolean parameters to indicate that the location of the WP can be moved to improve CPL.

TABLE X
AC ALGORITHM

Input: WP I, $LS(I, J)$, and $LS(I, K)$. Points I, J, and K are neighboring points with respect to the CP (See Fig. 17 for illustration).
Output: AC I, AC I will be used to find a single smooth curve segment.
Algorithm Steps Description:

1. Find the third leg, LS(J,K), to get a triangle.
2. Find a rectangle enclosing points I, J, and K.
3. Find a HLS or a VLS that pass through WP I and also intersect with LS(J,K).
4. Find an AC with center on the HLS or VLS in step 3. Set $ACD = 0.5min(LS(I,J), LS(I,K), LS(J,K))$ or user specified-value.

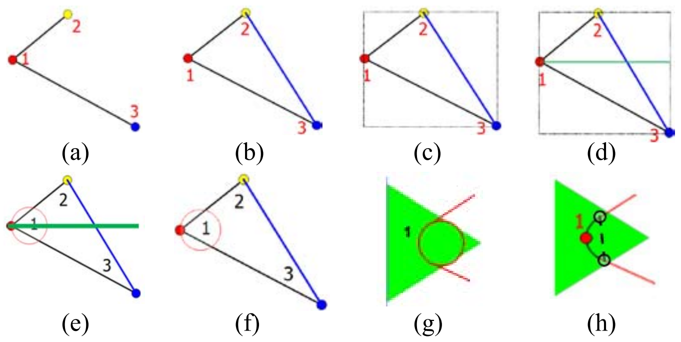
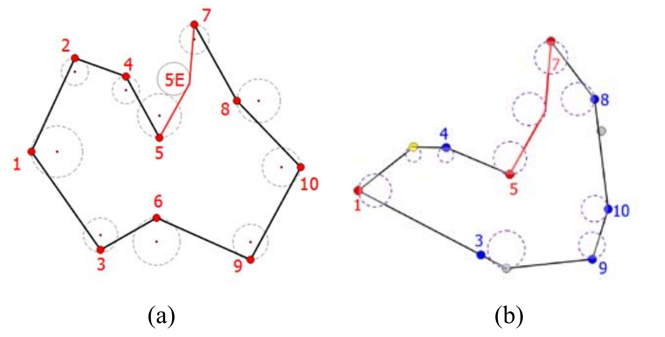


Fig. 17. ACC for input tour. (a) Two LSs. (b) Triangle of 3 WPs. (c) Enclosing rectangle. (d) HLS spawn by WP1. (e) AC versus HLS. (f) AC. (g) Tangent circle. (h) Desired CS.

D. Smooth Path Replanning

Applying adaptive curvature control (ACC) algorithm on all LSs of the CP, the smooth CP for the TR can be found. Fig. 17 illustrates the process of adaptive circle (AC) algorithm, one of two components of the ACC algorithm, which begin with the OP being considered and the two LSs connecting to it, LS(1,2) and LS(1,3). Fig. 17(a) is a segment of the CP shown in Fig. 15.

To find an AC which is proportional to the 2 LSs, LS(1,2) and LS(1,3), the segment is first transformed into a triangle by connecting the open endpoints in Fig. 17(a) to get Fig. 17(b). Then find the minimum enclosing rectangle to wrap around the triangle. The third base of the triangle, LS(2,3), is always the opposite length to the WP under consideration. After finding the triangle as shown in Fig. 17(b), the minimum enclosing rectangle maybe found with the three vertices of the triangle. The process of finding the MER is identical to that in Fig. 4(d). The vertical line test (VLT) and the horizontal line test (HLT) at WP1 will help determine the center of the detour AC which is required to be on the VLT or the HLT that intersect the boundary of the rectangle and the third base of the triangle. Fig. 17(d) illustrated the case that the HLT intersect



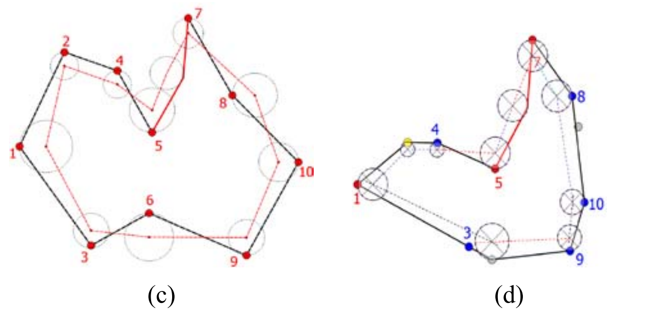


Fig. 18. ACs and their reference LSs. (a) AC and the CP. (b) AC and the PNWCC CP. (c) ACs and reference LSs. (d) ACs and reference LSs.

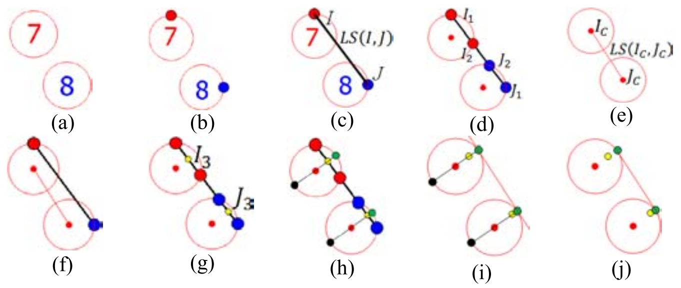


Fig. 19. Transition from ACs to desire tangent LS. (a) ACs. (b) WPs. (c) WP LS. (d) I_2 and J_2. (e) LS. (f) Two LSs. (g) I_3 and J_3. (h) I_b and I_g. (i) LP(I_t, J_t). (j) LS(I_t, J_t).

the boundary rectangle as well as the third base of the triangle

$$\text{ACD} = \frac{1}{2}\min(L(P_1, P_2), L(P_1, P_3), (P_2, P_3)). \qquad (14)$$

This can be visualize with the leftmost side of the rectangle that pass through WP1; it did not intersect the third base. Fig. 17(e) shows that the center of an AC is on the HLT as required. There may be two ACs to consider when both the HLT and the VLT meet the condition in step 3 of the AC algorithm. Either one may be used. Adaptive circle's diameter is equate in (14). It may also be set to a certain size for optimal condition depending on the vehicle and its operational requirements rather than with adaptive size.

Fig. 18(a) and (b) illustrates the piecewise continuous CP of the sample TR given in Fig. 2. They are the same as the CP in Figs. 9(b) and 15(b), respectively, except that the actual disks of the TR are not shown; only the CP and ACs are shown for clarity. The number of AC is equal to the number of WP, initially. Most of the WPs in Fig. 18 are OWs, except for the WP between WPs 5 and 7 which are collision avoidance WP. In Fig. 18(a), it is the WP closest to the enlarged disk 5 which is labeled as 5E. At this point, it should be clear that the CP obtain after the PNWCC algorithm reduced the CPL, reduce the number of observers, and also reduce the number of turns. LS(2,4) in Fig. 18(a) will be replaced with external tangent LS because it does not intersect with LS(AC2, AC4) as shown in Fig. 18(c). If the two were to be intersected, then LS(2,4) will be replaced with internal tangent LS. Fig. 19 illustrates the process of replacing the piecewise continuous LS for smoothness. Note that OWs are not replaced, they remain at the same location. Only the CP's LS leading to the OWs are replaced and linked to OW by a CS of the AC. See Fig. 20 for the complete transformation from piecewise continuous CP to the smooth CP and the preservation of the OWs.

Fig. 19 provided illustration for discontinuous to continuous (DC) transformation algorithm shown in Table XI. ACs 7 and 8 in Fig. 19(a) may be referred to as ACs I and J, respectively, in the DC algorithm. In this example, as well as in all of the WPs in Fig. 18, LS(7,8) presents the LS from points 7 to 8. LP(7,8) is a line passing through both points 7 and 8. LP(7,8) is longer than LS(7,8) and it contains all elements of LS(7,8). The concept of LP is required to generalize an LS that is not long enough to intersect a circle at two points when it could if it were longer. For example, LS(7,8) of Fig. 18(c) intersect AC 7 at two points and intersect AC 8 at only 1 point due to its length. The extension of LS(7,8) to LP(7,8) will intersected AC 7 at two points and AC 8 at two points. The LS(7,8) of Fig. 18(d) does not need to be extended to LP(7,8) as it is already intersecting AC 7 at two points and AC 8 at two points. The extension is necessary for some LSs to find a true tangent LS for smooth CP. Obvious LSs that required extension include LS(7,8), LS(7,5E), and LS(5, 5E) of Fig. 19(c).

A piecewise continuous CP is not desirable for nonholonomic robot. Before the CP can be transformed into a smooth and continuous CP with AC, the relationship between the two ACs and the piecewise continuous CP's LS must be known so that an internal or external tangent LS can be found. Fig. 18(a) and (b) must be transformed into Fig. 18(c) and (d), respectively. The difference between Fig. 18(a) and (c) is that Fig. 18(c) also connected an LS from center of two ACs that was connected with an LS of two WPs on the two ACs. The LS that connect two ACs' centers is called a reference LS. For example, WPs 1 and 2 are connected on two ACs, now both ACs are connected center to center to form a reference LS.

The relationship between LS(1,2) and LS(AC1, AC2) will require an external tangent LS modification and the tangent line is to the left of the reference LS. The relationship between LS(4,5) and the reference LS(AC4, AC5) will required an internal tangent LS modification. The PNWCC algorithm may alter the relationship between a WP LS and a reference LS. For example, LS(7,8) in Fig. 18(c) will require an internal tangent LS modification, but its counterpart in Fig. 18(d) will require an external tangent LS modification. After finding the WP's LS and reference LS relationship for all LSs, the smooth CP can be obtained as shown in Fig. 20 with the DC process illustrated in Fig. 19. While Fig. 19 shows only one iteration

TABLE XI
DC ALGORITHM

```
Input: A$LS(I,J)$, AC I, and AC J. See Fig. 19 (c) for illustration.
Output: New WPs $I_t$ and $J_t$ and new $LS(I_t,J_t)$. $I_t$ and $J_t$ are WPs on ACs I and J respectively. $LS(I_t,J_t)$ is tangent to AC I. Likewise, it is also tangent to AC J.
Algorithm Steps Description:
1.  Initialize $State\ I = False$ and $State\ J = False$. Compute $LP(I,J)$.
2.  Compute the intersection of $LP(I,J)$ and $AC\ I$, see Fig. 19(c) for illustration. If only one intersection is found, set $I_t = I$ and $State\ I = True$. Else if two intersections are found, set $I_1 = I$ and $I_2$ equal to the new point.
3.  Compute the intersection of $LP(I,J)$ and AC J. If only one intersection is found, set $J_t = J$ and $State\ J = True$. Else if two intersections are found, set $J_1 = J$ and $J_2$ equal to the new point.
4.  If both $State\ I = True$ and $State\ J = True$, end the program. Else continue to step 5.
5.  If $State\ I = False$, compute $I_3$, the average of $I_1$ and $I_2$, then compute $LP(I_C,I_3)$, see yellow points in AC I in Fig. 19(g) which represent $I_3$. Compute points $I_b$ and $I_g$ which are collinear to $LP(I_C,I_3)$ and intersect AC I. $I_b$ is for a black point and it is below LP($I_C,J_C$) and $I_g$ is for a green point and it is greater than LP($I_C,J_C$). $I_b$ and $I_g$ are on AC I. See Fig. 19(h).
6.  If $State\ J = False$, Compute $J_3$, the average of $J_1$ and $J_2$, then compute $LP(J_C,J_3)$, see yellow points in AC J in Fig. 19(g) which represent $J_3$. Compute points $J_b$ and $J_g$ which are collinear to $LP(J_C,J_3)$ and also intersect AC J. $J_b$ is for black point and it is below LP($I_C,J_C$) and $J_g$ is for a green point and it is greater than LP($I_C,J_C$). See Fig. 19(h).
7.  If I is less than LP($I_C,J_C$), set $I_t = I_b$. Else set $I_t = I_g$.
8.  If J is less than LP($I_C,J_C$), set $J_t = J_b$. Else set $J_t = J_g$.
9.  Compute the intersection of $LP(I_t,J_t)$ and AC I. If only one intersection is found, set $State\ I = True$. Else if two intersections are found, set $I_1 = I_t$ and $I_2$ equal to the new point.
10. Compute the intersection of $LP(I_t,J_t)$ and AC J. If only one intersection is found, set $State\ J = True$. Else if two intersections are found, set $J_1 = J_t$ and $J_2$ equal to the new point. Go to step 5.
11. If both $State\ I = True$ and $State\ J = True$, end the program.
```

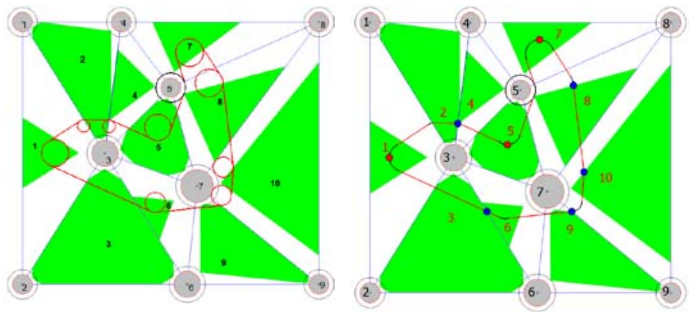


Fig. 20. First order differentiable CP due to AC's CS. (a) Smooth PNWCC CP. (b) Final smooth CP.

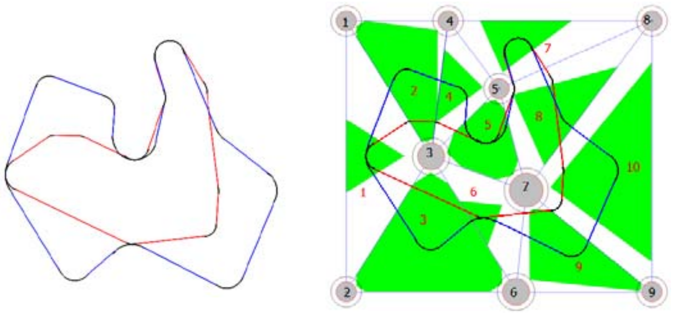
of DC transformation, some ACs and LSs in

$$s'(x) = L\big(p_{\pi(m')}, p_{\pi(1)}\big) + \sum_{i=1}^{m'-1} L\big(p_{\pi(i)}, p_{\pi(i+1)}\big) + \sum_{i=1}^{m'} \mathrm{arc}(i). \tag{15}$$

Fig. 18 may take up to three iterations to find the smooth CSs, but no more than three iterations. Each CS in Fig. 20(b) has three WPs. CSs 1, 4, 5, 7, 8, 10, 9, and 6 have 2 WPs and 1 OW. CS that detours around disk 5 has three WPs with no OW. The length of the CP shown in Fig. 20(b) can be found from (15). Note the fundamental difference between (11) and (15). All WPs in (11) are OWs, but not all WPs in (15) are OWs. Once all ACs and their WPs are found for a piecewise continuous CP, their desirable CSs as illustrated in Fig. 17(h) can be found by drawing an LS to connect the two tangent WPs, I_t and J_t, and then keep the CS that contains all three WPs. The smooth CP are obtained as in Fig. 21 by connecting all CSs and all tangential LSs.

TABLE XII
STATIC WORKSPACE CONFIGURATION IN ASSUMPTION 4

Disk	O_{ix}	O_{iy}	R_i	Disk	O_{ix}	O_{iy}	R_i
1	100	100	30	6	600	900	40
2	100	900	30	7	630	600	50
3	350	500	40	8	1000	100	30
4	400	100	30	9	1000	900	30
5	550	300	30				



(a) (b)

Fig. 21. Differentiable CP of the TR. (a) Centroid CP (blue) and PNWCC CP (red). (b) Centroid (blue) and PNWCC CPs (red).

V. SIMULATION

We simulate the data in Table XII with CGAL [15]. As seen throughout this paper, our algorithm works for both limited sensing range and infinite sensing range. Fig. 8 showed a limited sensing range in a single RT in the LOP's phase. Fig. 10 showed the baseline path planning (BPP)'s and path and observer replanning (POR)'s phases of a limited sensing range. For sufficient sensing range, Fig. 9 showed the LOP's phase, Fig. 15 showed the BPP's and POR's phases, and Fig. 20 showed the smooth path replanning's phase. All results are obtained in quadratic time due to the TSP's running time which is the slowest in all activities of the TBCPP approach.

A. Path Comparison

Fig. 21 shows two smooth CPs for the TR given in Fig. 2. The smooth CP in blue is the CP result from sufficient sensing range in each and every RT with their OWs being the centroid of the VPs of all RTs as shown in Fig. 6(b). WP 11 is the only collision avoidance WP. The smooth CP in red is the CP from Fig. 20(b). The red CP is much shorter than the blue CP. It is the result of the PNWCC Algorithm 2.

TABLE XIII
RUNNING TIME OF TRIANGULATION-BASED CP PLANNING

1	**Local Observer Planning**	
	Triangulation algorithm	N log N
	Visible Polygon algorithm	$O(M)$
	RBOP algorithm	$O(KM) \approx O(M)$
2	**Baseline Path Planning**	
	Traveling Salesman algorithm	$O(M^2)$
	Static Collision Avoidance	$O(M)$
3	**Path and Observers Replanning**	
	PNWCC	$O(M)$
	Static Collision Avoidance	$O(M)$
4	**Smooth Path Replanning**	
	Adaptive Curvature algorithm	$O(M)$
Summing all running time in all 4 phases		$O(M^2)$

B. Performance Comparison

It is obvious that the red CP has only eight observers while the blue CP has ten observers as shown in Fig. 21(b). For small sensing range as in Fig. 8, the initial number of observer is 11 while the improve CP has only 10 observers with reduced CPL.

C. Running Time and Convergence

A number of data structures are required to compute the running time of the TBCPP approach. They include the RT's and LS' data structures. Table XIII shows the running time of all four phases of our TBCPP approach with N number of disks and M number of RTs. The relationship between N and M can be found in (3), and M is generally slightly larger than N when there are more than three interior disks in the TR. For example, disks 3, 5, and 7 are interior disks for the given TR in Fig. 3. Due to these three interior disks, the number of RT is greater than the number of disks. In their absence, the number of RT is smaller than the number of disk. The number of LS as represents by E in (3) is related to N and M. k_i is a constant number greater than 1 for all values of i from 1 to M, but it may be set to 1 in complexity analysis in term of big-O notation [18]. ACC combines the AC algorithm and the DC algorithm to transform piecewise continuous CP into first-order differentiable CP and it accomplished this in linear time.

Visually, using RBOP algorithm to fill the necessary number of observers in each and every RT due to relatively smaller sensing range as compare to the size of the RT does not increase the running time of the algorithm due to the convention of the big-O notation. Applying the RBOP algorithm like Fig. 8 for all RTs in Fig. 3, the number of observers can be expressed as

$$k_1 + k_2 + k_3 + \cdots + k_M \leq KM \tag{16}$$

where K is a constant much larger than 1, but much less than M for a very large geographic region. Then the running time for the overall TBCPP can be tabulated in Table XIII as quadratic.

VI. CONCLUSION

Our TBCPP approach provides a smooth CP with many desirable traits such as complete and overlapped coverage control, curvature and path length control, localization, and choice of observer's placement based on sensing range. In addition, the algorithm also provides collision avoidance with static disks in the TR. Our family of algorithm begins by partitioning the TR into several RTs. Based on the size of the RT and the sensing range, the number and location of observers are determined for all RT. All observers found are then used as WPs to generate baseline CP with TSP's NN algorithm. Finally, the last algorithm of our approach process all segment of the CP to find smooth curvature that preserve the location of the observers. The overall CPL is very easy to compute as it consists of LSs and circles' CSs. Our result can be integrated with recent results [22]–[25] to obtain optimal, robust, and low cost system and products that improve our quality of life.

REFERENCES

[1] V. An and Z. Qu, "Triangulation-based path planning for patrolling by a mobile robot," in *Proc. Aust. Control Conf.*, Fremantle, WA, Australia, 2013, pp. 183–188.

[2] V. An and Z. Qu, "A practical approach to coverage control for multiple mobile robots with a circular sensing range," in *Proc. Robot. Sensors Environ.*, Washington, DC, USA, 2013, pp. 112–117.

[3] C.-H. Kuo, H.-C. Chou, and S.-Y. Tasi, "Pneumatic sensor: A complete coverage improvement approach for robotic cleaners," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 4, pp. 1237–1256, Apr. 2011.

[4] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Proc. Int. Conf. Field Service Robot.*, Canberra, ACT, Australia, 1997, pp. 203–209.

[5] H. Choset, "Coverage for robotics—A survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, nos. 1–4, pp. 113–126, 2001.

[6] T.-K. Lee, S.-H. Baek, Y.-H. Choi, and S.-Y. Oh, "Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation," *Robot. Auton. Syst.*, vol. 59, no. 10, pp. 801–812, 2011.

[7] J. S. Oh, Y. H. Choi, J. B. Park, and Y. F. Zheng, "Complete coverage navigation of cleaning robots using triangular-cell-based map," *IEEE Trans. Ind. Electron.*, vol. 51, no. 3, pp. 718–726, Jun. 2004.

[8] Y. Gabriely and E. Rimon, "Competitive on-line coverage of grid environments by a mobile robot," *Comput. Geometry*, vol. 24, no. 3, pp. 197–224, 2003.

[9] A. Yazici, G. Kirlik, O. Parlaktuna, and A. Sipahioglu, "A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints," *IEEE Trans. Cybern.*, vol. 44, no. 3, pp. 305–314, Mar. 2014.

[10] L. Paull, C. Thibault, A. Nagaty, M. Seto, and H. Li, "Sensor-driven area coverage for an autonomous fixed-wing unmanned aerial vehicle," *IEEE Trans. Cybern.*, vol. 44, no. 9, pp. 1605–1618, Sep. 2014.

[11] N. L. Doh, C. Kim, and W. K. Chung, "A practical path planner for the robotic vacuum cleaner in rectilinear environments," *IEEE Trans. Consum. Electron.*, vol. 53, no. 2, pp. 519–527, May 2007.

[12] M.-C. Kang, K.-S. Kim, D.-K. Noh, J.-W. Han, and S.-J. Ko, "A robust obstacle detection method for robotic vacuum cleaners," *IEEE Trans. Consum. Electron.*, vol. 60, no. 4, pp. 587–595, Nov. 2014.

[13] R. Goroshin, Q. Huynh, and H.-M. Zhou, "Approximate solutions to several visibility optimization problems," *Commun. Math. Sci.*, vol. 9, no. 2, pp. 535–550, 2011.

[14] Z. Qu, J. Wang, and C. E. Plaisted, "A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles," *IEEE Trans. Robot.*, vol. 20, no. 6, pp. 978–993, Dec. 2004.

[15] CGAL. *Computational Geometry Algorithms Library*. Accessed: Jan. 2013. [Online]. Available: http://www.cgal.org

[16] C. Nilsson, "Heuristics for the traveling salesman problem," Dept. Comput. Inf. Sci., Linköping Univ., Linköping, Sweden, Rep., 2003.

[17] C. D. Kozen, *The Design and Analysis of Algorithms*. New York, NY, USA: Springer, 2012.

[18] M. T. Goodrich, R. Tamassia, and D. M. Mount, *Data Structures and Algorithms in C++*, 2nd ed. Danvers, MA, USA: Wiley, 2009, pp. 594–654.

[19] K. T. Leung and S. N. Suen, *Vectors, Matrices and Geometry*, vol. 1. Hong Kong: Hong Kong Univ. Press, 1994.

[20] J. O'Rourke, "Polygon partitions," in *Art Gallery Theorems and Algorithms*. New York, NY, USA: Oxford Univ. Press, 1987, pp. 1–29.

[21] S. Arora, “Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems,” *J. ACM*, vol. 45, no. 5, pp. 753–782, Sep. 1998.

[22] C. Chen, Z. Liu, Y. Zhang, C. L. P. Chen, and S. Xie, “Saturated Nussbaum function based approach for robotic systems with unknown actuator dynamics,” *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2311–2322, Oct. 2016.

[23] G. Lai, Z. Liu, Y. Zhang, and C. L. P. Chen, “Adaptive fuzzy tracking control of nonlinear systems with asymmetric actuator backlash based on a new smooth inverse,” *IEEE Trans. Cybern.*, vol. 46, no. 6, pp. 1250–1262, Jun. 2016.

[24] V. An, Z. Qu, and R. Roberts, “A rainbow coverage path planning for a patrolling mobile robot with circular sensing range,” *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[25] V. An and M. Soares, “A low cost technique to evaluate usable product for small manufacturing companies: A case study on Garcia robot,” *Work*, vol. 41, no. S1, pp. 2089–2096, 2012.

Vatana An (M’13) received the bachelor’s degrees in electrical engineering and computer engineering from the University of Florida, Gainesville, FL, USA, in 2002 and the master’s and Ph.D. degrees in electrical engineering from the University of Central Florida, Orlando, FL, USA, in 2005 and 2017, respectively.

His current research interests include networked systems, cooperative controls, robotics, and autonomous vehicle systems.

Zhihua Qu (M’90–SM’93–F’10) received the Ph.D. degree in electrical engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 1990.

Since then, he has been with the University of Central Florida, Orlando, where he is currently a Professor with the Department of Electrical and Computer Engineering. He has published a number of papers in electrical and computer engineering. He has authored three books.

Frank Crosby (M’13) received the Ph.D. degree in mathematics from the University of Florida, Gainesville, FL, USA, in 1995.

He joined the Naval Surface Warfare Center, Panama City Division, Upper Grand Lagoon, FL, USA. He has also a Visiting Scholar appointment from the Georgia Institute of Technology, Atlanta, GA, USA. He was researching in multispectral imaging and granted a patent for an automatic target recognition algorithm.

Dr. Crosby was a recipient of the American Society for Photogrammetry and Remote Sensing Award for Best Scientific Paper in Remote Sensing for one of his papers titled “Adaptive Correlation Analysis With Non-Overlapping Imagery Indication” in 2008.

Rodney Roberts (M’91–SM’02) received the B.S. degrees in electrical engineering and mathematics from the Rose-Hulman Institute of Technology, Terre Haute, IN, USA, in 1987 and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1988 and 1992, respectively.

From 1992 to 1994, he was a National Research Council Fellow with Wright Patterson Air Force Base, Dayton, OH, USA. Since 1994, he has been with the Florida A&M University–Florida State University College of Engineering, Tallahassee, FL, USA, where he is currently a Professor of Electrical and Computer Engineering. His current research interests include robotics and image processing.

Vithia An (M’12) received the M.S. degree in biology from Stanford University, Stanford, CA, USA, in 2012 and the master’s degree in computer science from the University of West Florida, Pensacola, FL, USA, in 2017.

Her current research interests include biological sciences and computer science.