A Rainbow Coverage Path Planning for a Patrolling Mobile Robot With Circular Sensing Range

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

***Abstract*—In this paper, we proposed a coverage path (CP) planning approach for a mobile robot moving in a dynamically changing environment. Our family of algorithms begins by finding the first path around the center of interior obstacles/disks of a given target region (TR) through the Graham scan algorithm. The first path is then used as a foundational path to generate a collision free, first order differentiable, and observable CP through a complete set of input and output transformation algorithms which together form the rainbow CP planning approach. The last algorithm of the rainbow CP planning approach finds a sufficient number of observation points on the CP needed to observe the TR. The novelty of our rainbow CP planning approach is that it partitions the TR into different shapes with different properties needed to obtain complete coverage while achieving first order path differentiability. The main technical contributions of the proposed approach is to provide a holistic solution that segments any TR, uses triangulation to determine the line of sights and observation points, and computes the collision-free CP within a quadratic runtime. The proposed method can be readily generalized to address problems of higher dimensions, and it is scalable with respect to the size of TR and the number of robots. Computer simulations are used to illustrate the effectiveness and correctness of the proposed approach.**

***Index Terms*—Convex hull, dynamic, external tangent, half-plane, nonholonomic, triangulation, visible polygon (VP).**

TABLE I
COMMON ABBREVIATION (ABB.)

ABB.	Long form
B, C, G, R, and Y	Symbols for color representing blue, cyan, gray, red, and yellow respectively.
BCS	Blind Curve Segment
CET	Common External Tangent
CIT	Common Internal Tangent
CP	Coverage Path
CS	Curve Segment
CW	Clockwise
DT	Delaunay Triangulation
HLT	Horizontal Line Test
LS	Line Segment
OP	Observer Placement
BURL	Bottom, Upper, Right, and Left
CS(A,B)	Curve Segment Connecting Vertices A and B
CSIRT	Curve Segment Intersecting Regular Triangulation
CVCS	Candidate Visible Curve Segment
CWCP	Circular Waypoint Coverage Placement
L(A,B)	Line segment connecting points A and B or $\overline{AB}$
L(A,B,r)	LS connecting points A and B is a Red LS
LosPoRT	Line of sight Partition of Regular Triangulation
MER	Minimum Enveloping Rectangle
PNWCC	Previous Next Waypoint Coverage Constraint
RT	Regular Triangulation
SR	Sub Region
TR	Target Region
VC	Visible Circle
VCPM	Visible Circle Path Modification
VCS	Visible Curve Segment
VD	Visible Disk
VL	Vertical Line
VLS	Visibility Line Segment
VLT	Vertical Line Test
VP	Visible Polygon
WRT	With Respect To
XPath	RedPath, OrangePath, YellowPath, GreenPath, BluePath, IndigoPath, and VioletPath if X is R, O, Y, G, B, I, and V respectively.

I. INTRODUCTION

COVERAGE path (CP) planning for mobile robot to find a motion path for the robot to pass over all points in a given region has recently become an increasingly popular research topic. CP determines power usage that arises from massive information received from sensory devices, distance to travel and time to completion, and number of turns for the robot to cover the interested region [1]–[14]. The type of coverage such as approximate coverage or complete coverage is an important factor in CP planning.

Choset [4] defined four different types of coverage: 1) heuristic; 2) approximate; 3) partial-approximate; and 4) exact decomposition. Any type of cellular decomposition divides the interest region into "simple" cells. In any CP planning algorithm, complexity is usually measured with two different parameters. They are distance traveled by the robot and memory required to store input information. Table I lists

Manuscript received October 25, 2016; accepted January 21, 2017. Date of publication February 15, 2017; date of current version July 17, 2018. This work was supported by the U.S. Navy's Internal Laboratory Independent Research Grant. The work of Z. Qu was supported in part by the U.S. National Science Foundation under grant ECCS-1308928, in part by the U.S. Department of Energy under Award DE-EE0006340 and Award DE-EE0007327, in part by the U.S. Department of Transportation under Grant DTRT13-G-UTC51, in part by L-3 Communication under Contract 11013I2034, in part by Leidos under Contract P010161530, and in part by the Texas Instruments Awards. This paper was recommended by Associate Editor Z. Liu.

V. An is with the U.S. Navy, Panama City, FL 32407 USA, and also with the University of Central Florida, Orlando, FL 32816 USA (e-mail: vatana.an@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—Florida State University College of Engineering, Tallahassee, FL 32310-6046 USA (e-mail: rroberts@eng.fsu.edu).

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.2017.2662623

common acronyms that will be used throughout this paper. These acronyms will be defined at first usage.

Kinematic and boundary constraints are also major problems that have puzzled many researchers for decades. Efficient mobile coverage requires the path to be smooth or differentiable while avoiding obstacles. An important question to answer is how a coupled issue such as path differentiability and a complete coverage is obtained? In the span of one decade, [1], [2], [5], [6], and [12] achieved some degrees of success with CP planning. However, some fundamental issues need to be addressed. An and Qu [1] proposed a CP planning with triangulation technique which guarantees complete coverage, but as the sensing range approaches infinity, the final CP obtained is formed by many ripples. An and Qu [2] proposed a CP planning algorithm that improves the result of [1] while maintaining complete coverage; however, as the sensing range approaches infinity, saturation occurs. Oh *et al.* [6] proposed a triangular-cell-based map CP planning; however, the resulting CP is discontinuous. Yazici *et al.* [12] proposed a generalized Voronoi diagram-based CP planning; however, the resulting CP is also discontinuous. Lee *et al.* [5] attempted to smooth the path with a high-resolution grid map representation and coarse to fine constrained inverse distance transform techniques; however, discontinuity in the CP still exists.

Based on the issues discussed above, the problem statement is to design a differentiable CP that is as short as possible as well as obtaining complete coverage when sufficient sensing range is available. Our new result, unlike that in [5], [6], [12], and [14], provides a smooth CP. Because motion control points can be selected to be on the smooth CP, smooth speed adjustment, and smooth steering is also possible due to the result of [15]. Unlike our previous results in [1] and [2] which compute all observer points or observers first and then the differentiable path second, the result of this technique is the opposite. The reason for inefficient CP in [1] and [2] are due to the locality consideration. What this mean is that only a few disks in the set are considered when observers are found which implies that only a few disks in the set are considered when CP is designed. This is different from global consideration where all disks in the set are considered in implementing the CP which is implemented in this paper. Our new contribution is the rainbow CP planning approach which requires seven phases to obtain a smooth and differentiable CP with sufficient observers needed to observe the target region (TR). The rainbow CP planning algorithm will be referred to as the rainbow algorithm for brevity. Most of the algorithms in this paper are our new contributions. Some existing algorithms that we incorporated will be mentioned wherever they are introduced. Finally, the overall organization of this paper is as follows. Section I-A describes some examples of exact decomposition. Sections I-B and I-C describe the main tools to implement a piecewise continuous control law and collision avoidance with dynamic obstacles. In Section II, we formulate our problem and define the necessary assumptions. In Section III, we discuss existing and newly developed algorithms. We propose the rainbow algorithm, in Section IV. We verify and validate our techniques in Section V and then provide our conclusion in Section VI.

A. Exact Decomposition

An exact cellular decomposition technique results in the set of nonintersecting regions and their union is the region of interest [4]. Examples of exact cellular decomposition include trapezoidal, boustrophedon [3], triangular, and rectangular decomposition. Our CP planning approach for a robot uses a triangular decomposition approach. CP planning for multiple robots takes the coverage control into consideration. A triangular decomposition is preferred because of its simplicity and because theories and algorithms are available to solve it such as Delaunay triangulation (DT) [16] and the art gallery theorem [17].

B. Canonical Control

Canonical chained form allows a nonholonomic system in the world coordinate to be transformed into a controllable form. The controllable inputs are piecewise continuous which naturally fit the main result of this paper. Chained form examples can be found in [15] and [18].

C. Obstacle Avoidance and Trajectory Generation

Recent advances in trajectory generation allow for a real-time path planning for a whole class of nonholonomic system. Qu *et al.* [15] formulated and proved an input parameterization approach which is analytical and completely controls an autonomous system in a piecewise continuous manner. The result is practical because the robot can be analytically controlled to move from an initial position to a final position within the time, kinematic, and geometrical constraints. The result of [15] is improved in [18] with regard to near-optimal path length and control energy.

II. Problem Formulation

This section defines 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 2-D obstacle, with the center at $O(t) = (x, y)$ and of radius R_r. Its motion obeys a nonholonomic constraint with velocity vector expressed as $v_r(t)$. The position and velocity are a function of time because the robot is continuously moving.

Assumption 2: The radius or range of the robot's motion sensor is R_s. R_s is greater than R_r and has sufficient length to observe TR(s) from an observation point.

Assumption 3: Both type of objects, static and dynamic, are denoted by the symbol $O_i(t)$, where the subscript $i = 1, \ldots, N$ represents the obstacle number. For example, an ith object with radius R_i will be represented by obstacle centered at point $O_i(t)$. For moving objects, the origin $O_i(t)$ is time varying and moves with piecewise linear velocity.

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 TR in Fig. 1, where disks 1–4, 7, 13, 14, and 16–18 are exterior while disks 5, 6, 8–12, and 15 are interior. It is assumed that moving obstacles are moving within the TR.

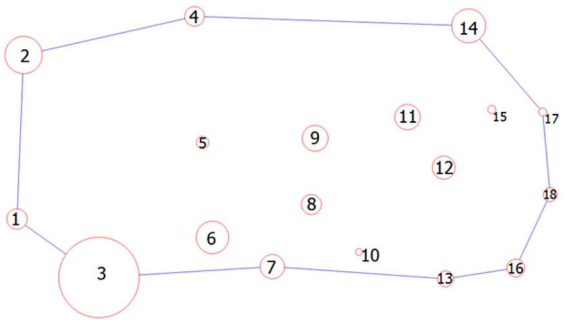


Fig. 1. Sample TR.

A. Patrolling Control Problem

The problem is to design a differentiable and continuous path for a nonholonomic mobile sensing robot with range, coverage, movement, and time constraints to 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 piecewise, continuous steering control under which the robot moves collision-free, and covers all points in the set Ω overtime. Mathematically, the problem is to determine a differentiable path $s(t)$ by ensuring conditions represented by (1) and (2) hold

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

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

where q is the search space, $\phi(q,t)$ is a weighing function according to constraints and other design requirements, and T is the time for the robot to complete its search mission. t_0 is an initial time. Equation (1) is a standard mathematical expression for CP planning, and inequality (2) is a mathematical representation of no collision. It is shown in [19] that the choices of $\phi(q,t)$ may produce $s(t)$ that passes through a given point q once or several times and at certain desirable time. The patrolling control problem is solved using the proposed rainbow CP planning approach which consists of the sets of algorithms tabulated in Tables II and III.

The rainbow algorithm requires a number of seed algorithms which includes some existing algorithms and newly developed algorithms. All 13 algorithms will be explained in the subsequent sections. The existing guidance and control algorithm from [15] is also included in Section V-A for completeness.

III. Coverage Strategies

In this section, we discuss auxiliary algorithms presented in Table III. These algorithms use the following six known results.

1) *Minimum Enveloping Rectangle Algorithm:* Minimum enveloping rectangle (MER) algorithm takes a finite number of vertices as input and outputs the vertices with extreme coordinates $(x_{\min}, y_{\min})$, $(x_{\min}, y_{\max})$, $(x_{\max}, y_{\min})$, and $(x_{\max}, y_{\max})$ [20]. With an example convex polygon shown in Fig. 2(a) and (d), MER algorithm returns four extreme vertices that form a simple MER as shown in Fig. 2(b) and (e), respectively, in dashed. From a MER, a finite length line segment (LS), vertical and horizontal, with sufficient length can be found to test relative CSs that form a closed path as shown in Fig. 2(c) and (f). The closed path in Fig. 2 consists of four connected CSs and their endpoints are known as vertices.

2) *Common External Tangent and Common Internal Tangent Algorithm:* Common external tangent (CET) and common internal tangent (CIT) algorithm finds an LS that is tangent to two coplanar circles that does not intersect or does intersect the segment joining the centers

TABLE II
Seven Phases of the Rainbow Algorithm and Their Descriptions

Find a foundational path as a function of the TR
1. RedPath algorithm - Find a convex hull or foundational path from interior disks' centers.
Find a collision free path that inherit the foundational path
2. OrangePath algorithm – Modifies the RedPath to get a CP that envelopes all interior disks.
3. YellowPath algorithm – Modified the OrangePath as necessary to get a CP that envelopes all interior disks and is collision free with all disks in the TR.
Find a collision free path that can observe all crossing Regular Triangulations (RT)
4. GreenPath algorithm – Modifies the YellowPath as necessary to get a CP that can observe all crossing RTs of the TR.
Find a differentiable collision free path that can observe all crossing RTs
5. BluePath algorithm – Modifies the GreenPath as necessary to get a first order differentiable CP.
Find a differentiable collision free path that can observe all crossing and non crossing RTs
6. IndigoPath algorithm – Find Visible Curve Segments (VCSs) on the BluePath that can observe all noncrossing RTs and noncrossing SRs.
7. VioletPath algorithm – Find a sufficient number of observers on the BluePath to observe the TR.

TABLE III
Auxiliary Algorithms Used by the Rainbow Algorithm

1. BURL algorithm – A relative Curve Segment (CS) test developed for OrangePath's, YellowPath's, GreenPath's, BluePath's, and IndigoPath's algorithms.
2. CSIRT algorithm - Developed for the GreenPath algorithm to find observers on the CP for certain RT.
3. VCPM algorithm - Developed for the GreenPath algorithm to modify part of the YellowPath to obtain complete coverage of the TR.
4. LosPoRT algorithms –Developed for the GreenPath, BluePath, and an IndigoPath algorithms to find VCS on the CP for noncrossing RT or noncrossing SR. There are three variants. LosPoRT 1 and 3 partitions an RT into 3 sub regions. LosPoRT 2 partitions LosPoRT sub region into smaller regions.
5. OP algorithm–Developed for the VioletPath algorithm to find sufficient number of observers for the TR.

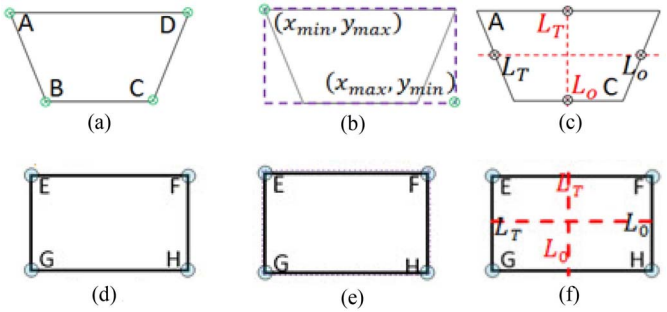


Fig. 2. Sample MER and relative CS test. The CSs in (a)–(c) are $\overline{AD}$, $\overline{DC}$, $\overline{BC}$, and $\overline{AB}$. The CSs in (d)–(f) are $\overline{EG}$, $\overline{EF}$, $\overline{HF}$, and $\overline{HG}$. (a) Typical closed path. (b) and (e) MER. (c) and (f) Relative LS test. (d) Closed path.

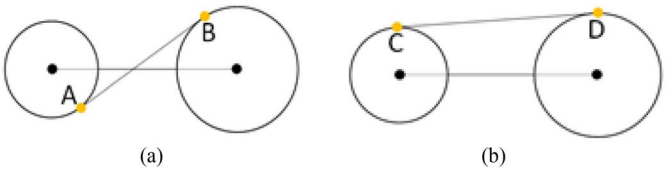


Fig. 3. Examples illustrating CIT and CET. (a) $L(A, B)$ is a CIT. (b) $L(C, D)$ is a CET.

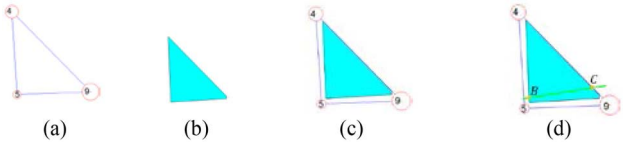


Fig. 4. Example of an RT, a VP, and a CS crossing both. (a) RT 6. (b) VP 6. (c) RT 6 and VP 6. (d) $CS(B, C)$.

of the two circles, respectively [21]. Fig. 3 illustrates a CIT/CET with $\overline{AB}$ and $\overline{CD}$, respectively.

3) *Visible Polygon Algorithm:* Visible polygon (VP) algorithm takes a regular triangulation (RT) as input and outputs its VP [1]. The shape and size of a VP is a function of the RT. An and Qu [1] guaranteed that if an observer of sufficient range is placed inside of an RT's VP, then the entire RT is observable. The circular waypoint coverage placement (CWCP) algorithm places an observer at the centroid of a VP to observe an RT. An and Qu [1] showed that a VP is always a convex polygon. The centroid of a convex polygon can be computed by the following equations, where A represents an area of the convex polygon, x_i and y_i represent vertices of the convex polygon, and C_x and C_y represent coordinates of the centroid [22]:

$$A = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1}y_i)$$

$$C_x = \frac{1}{6A}\sum_{i=0}^{n-1}(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1}y_i)$$

$$C_y = \frac{1}{6A}\sum_{i=0}^{n-1}(y_i + y_{i+1})(x_i y_{i+1} - x_{i+1}y_i).$$

In the above equations, n represents the number of vertices of the VP which has a maximum of nine vertices [1]. Fig. 4(a) shows one of the RTs of the TR, RT 6. Fig. 4(b) shows VP of RT 6. Fig. 4(c) shows the VP overlay on its RT. Fig. 4(d) shows an RT with its VP being crossed by a CS, $\overline{BC}$. Any sufficient-range observer on $\overline{BC}$ can observe RT 6. Equivalently, $\overline{BC}$ is the VCS of RT 6.

4) *Graham Scan Algorithm:* Graham scan algorithm finds convex hull of a finite set of points [23]. The convex hull of a set of N points in the Euclidean plane is the smallest convex set that contains all points in the set. Graham scan algorithm is implemented in the RedPath algorithm. Graham scan algorithm is generalizable to higher dimension.

5) *DT Algorithm:* DT algorithm partitions the TR into several RTs [16]. It is an exact partition technique implemented in the GreenPath algorithm. An RT is similar to a triangle with three edges and three vertices, except that the three vertices in a triangle are replaced by three circles of varying radii.

6) *Line Intersection Algorithm:* Line intersection algorithm computes the intersection between two or more CSs [24]. Line intersection algorithm is implemented in the VioletPath algorithm to find observers for the TR.

TABLE IV
RELATIVE CURVE SEGMENT TEST (BURL) ALGORITHM

Input: A closed path.
Output: Relative status of each CS in a closed path.
Algorithm Steps Description:
1. Find a MER for a closed path and determine VLT and HLT.
2. Find the midpoint of each CS.
3. Draw a VLT at the midpoint of each CS as shown in Fig. 2(c) and then compute L_0 and L_T. Determine the relationship between L_0 and L_T. Get the CS status as either "upper" or "bottom". If L_0 is not found or the slope of the CS is undefined, go to step 4. Else return status of the CS.
4. Draw a HLT at the midpoint of each CS then compute L_0 and L_T. Determine the relationship between L_0 and L_T. Get the CS status as either "left" or "right". Return status of the CS.

Based on the aforementioned algorithms, the auxiliary algorithms, defined and explained below are used to facilitate the implementation and integration of the rainbow algorithms.

A. Auxiliary Algorithm 1 (BURL Algorithm)

The bottom, upper, right, and left (BURL) algorithm is developed to determine the closed path's CS status. It consists of four steps to form a relative CS test algorithm as shown in Table IV. To handle all cases of CSs in a closed path, both vertical line test (VLT) and horizontal line test (HLT) must be considered. For example in Fig. 2(c), the VLT tests $\overline{AD}$'s midpoint for two intersection points, L_T and L_O. Likewise, the HLT tests $\overline{AB}$'s midpoint for two intersection points, L_T and L_O. If the CS under test is a bottom/upper CS, then L_T is expected to be below/above L_O. Since $\overline{AD}$ is an upper CS, L_T is above L_O. If the CS under test is a left/right CS, then L_T is expected to be left/right of L_O. Fig. 2(f) illustrates the case of VLT on $\overline{GE}$ in which the status of $\overline{GE}$ cannot be determined. All CSs are considered in clockwise (CW) direction in this paper. VLT in Fig. 2(f) intersects $\overline{GE}$ at L_T and infinitely many other points (no unique solution), and L_O cannot be determined. However, HLT determines that $\overline{GE}$ is a "left" CS. $\overline{GE}$ has undefined slope because it is a vertical line (VL).

B. Auxiliary Algorithm 2 (CSIRT Algorithm)

A curve segment intersecting RT (CSIRT) algorithm is developed to observe an RT from an arbitrary CS that crosses the RT, but does not cross the RT's VP. Fig. 5(b) illustrates the RT and CS pair that require CSIRT algorithm for observability. The VP in Fig. 5(a) is highlighted in the black circle on the longest edge of the RT. It is really tiny and it is enlarged in Fig. 6(b) as the convex region bounded by two green LSs and a blue LS. Every RT has six extreme points (EPs). A CSIRT algorithm partitions the RT along the visible disk (VD), the disk that has both visibility LSs (VLSs) intersect with the CS within an RT.

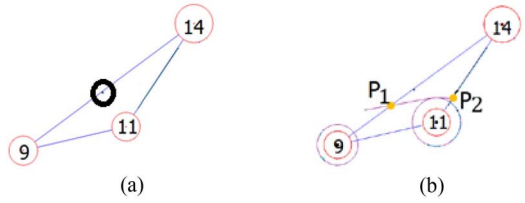


Fig. 5. RT 16, its VP, and an intersecting CS in RT. VP of RT 16 is enlarged in Fig. 6(b) which is also defined as A_4. (a) RT 16 and its tiny VP. (b) RT, VP, and CS(P_1, P_2).

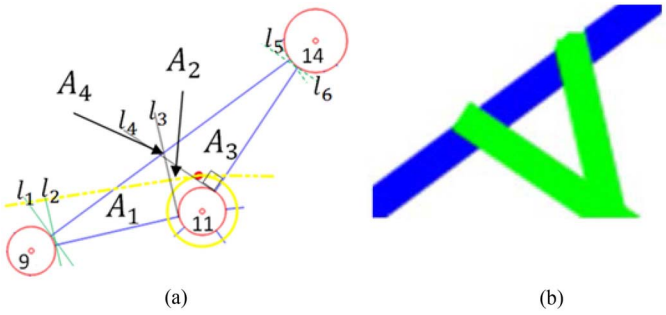


Fig. 6. RT through CSIRT algorithm. Disk 11 has two VLSs intersecting with the yellow CS. CSIRT only requires the yellow CS that is in an RT. Yellow CS is a CS of yellow path to be discussed later. Note that VLS is perpendicular to the RT's boundary LS that the EP is originated. (a) CSIRT partition of an RT. (b) Snapshot of A_4.

The algorithm begins by computing the VLSs of the RT. Of the three disks within the RT, only disk 11 has both VLSs intersect the curve segment of the CP as shown in Fig. 6(a). Lemma 1 guaranteed that if an RT can be partitioned by the CSIRT algorithm, then there exist three observers on the CS to observe an RT. The number of observers produced by the CSIRT algorithm is reduced to 2 by Theorem 1.

The CSIRT algorithm partitions the RT with a total area A into four mutually disjoint subregions (SRs), $A_1, A_2, A_3,$ and A_4, where the total area $A = \cup_{l=1}^{4} A_l$ and $|A| = \sum_{l=1}^{4} |A_l|$. The seven-step CSIRT algorithm is illustrated in Table V. A CSIRT algorithm partitions an RT into SRs and then finds observers on the CS to observe each SR. Lemma 1 is developed to guarantee that if an RT and the CS that cross the RT are partitionable pairs, then the RT can be observed by three observers. To formulate Lemma 1, let us define some definitions.

A slicing point is an auxiliary point on the VD [see Fig. 7(a)] that serves as a marker point of a SR in region A_2 [see Fig. 6(a)]. A slicing point I_1 is located between an observer O_1 and the center point of the VD. Likewise a slicing point I_2 is located between an observer O_2 and the center of the VD. A slicing LS is an auxiliary line of sight segment within the RT that serves to designate a SR of region A_2. There are two variants of slicing LSs. The first variant is the center slicing LS. A center slicing LS passes through an observer, a slicing point, and ends at the center of the VD. A center slicing LS l_1 has collinear points O_1, I_1, and the center point of the VD. Likewise a center slicing LS l_2 has collinear points O_2, I_2, and the center point of the VD. The second variant is the secant slicing LS. A secant slicing LS l_3 passes through an observer O_3 and I_1. A secant slicing LS l_4 passes through an observer O_3 and I_2. Now we are ready to present Lemma 1.

TABLE V
CSIRT ALGORITHM

Input: An RT and CS pair (eg. RT16 and CS(RT16)).
Output: Observers O_1, O_2, and O_3 on the input CS and mutually disjoined SRs which together form an RT.
Algorithm Steps Description:
1. Compute the VLSs l_i, $i \in 1,2,\ldots,6$ as shown in Fig. 6(a).
2. Find a VD. Return false if not found.
3. Check whether the two VLSs of every disk intersect with each other within the RT.
4. Determine SRs $A_1, A_2, A_3,$ and A_4 based on step 3's result.
5. In CW direction, find the first VLS of the VD that intersects with CS within the RT and designate the intersection point as O_1.
6. Repeat step 5 to find the second VLS that intersects with CS within the RT and designate the intersection point as O_2.
7. Compute the average x-value of O_1 and O_2 then find the corresponding y-value on the CS. This is O_3. Return true.

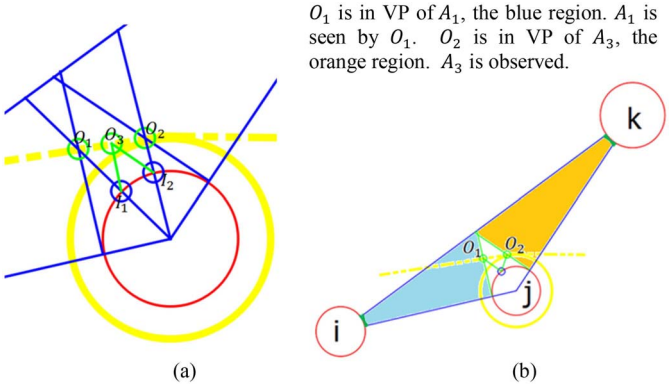


Fig. 7. Illustration of (a) Lemma 1 and (b) Theorem 1. Condition 1 returns observers O_1, O_2, and O_3. Condition 3 is shown with slicing points I_1 and I_2 connected to O_1 and O_2, respectively, and points I_1 and I_2 are connected to point O_3.

Lemma 1: The CSIRT algorithm provides a set of observers, O_1, O_2, and O_3, on a CS that crosses an RT, but does not cross its VP, that jointly observe the whole RT if all of the following conditions hold.

1) The CSIRT algorithm returns true with an RT and the CS pair as inputs.
2) O_1 is in the VP of A_1. Likewise O_2 is in the VP of A_3.
3) O_1, O_2, and O_3 have the following properties.
   a) A center slicing $\overline{O_1 I_1}$ intersects with a secant slicing $\overline{O_3 I_1}$ at I_1.
   b) A center slicing $\overline{O_2 I_2}$ intersects with a secant slicing $\overline{O_3 I_2}$ at I_2.

Proof: Condition 1 of the lemma guarantees that the combination of the input CS and an RT pair, partitions an RT into SRs with all three observers and all four SRs with properties needed for visibility analysis. Condition 2 of the lemma guarantees that O_1 has full coverage of A_1, and O_2 has full coverage of A_3. Condition 3 guarantees that O_1, O_2, and O_3 can jointly observe A_2. A_4, if greater than zero, is always seen by either O_1, O_2, or O_3 because it is not blocked. ■

Theorem 1 allows the number of observers to be reduced to 2 if a common point on the VD can be observed by the first two observers. Fig. 7(b) illustrates how Theorem 1 is applied.

Theorem 1: An RT that can be partitioned by the CSIRT algorithm and is visible by a set of observers can be reduced to just two observers if observers O_1 and O_2 can both observe a common point on the VD.

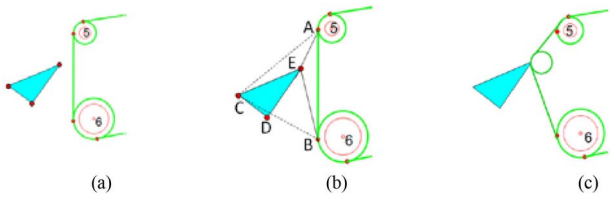


Fig. 8. VCPM algorithm for a VP to be crossed by the CP. (a) Scenario for CET. (b) Comparing distance. (c) VC and new CS.

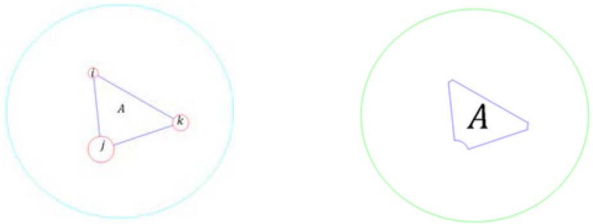


Fig. 9. RT A, with and without disks, to be observed from a CP $P(A)$. (a) RT forms by disks $i, j,$ and k. (b) Diskless region A is visible from every point on $P(A)$.

Proof: O_1 can observe A_1 because it is in the VP of A_1. Likewise, O_2 can observe A_3. The fact that O_1 and O_2 can both observe a common point on the VD means that together they can observe A_2. Since A_4 is not blocked, either O_1 or O_2 can observe A_4. ■

C. Auxiliary Algorithm 3 (VCPM Algorithm)

Visible circle path modification (VCPM) algorithm is developed to replace an LS that cannot observe a certain region with a CS that can observe the region. VCPM requires either the CET or CIT algorithm to find replacement CS depending on the location of the VP and the status of the LS to be replaced (e.g., left, right, etc.). Fig. 8 shows a scenario where VCPM algorithm is required. The LS's and VP's relative position with respect to each other can be checked by the BURL algorithm to compute the replacement CS. Distance of each vertex of the VP from the LS's endpoints can be compared as shown in Fig. 8(b) to find the shortest distance for insertion of visible circle (VC). Once VC is found, CET/CIT is then employed to find the replacement CS as shown in Fig. 8(c).

D. Auxiliary Algorithm 4 (LosPoRT 1 Algorithm)

Line of sight partition of regular triangulation (LosPoRT) 1 algorithm is developed to observe an RT that the CP does not cross. Suppose we are given a CP *P(A)* surrounding an RT *A* as shown in Fig. 9(a). It is the simplest example of LosPoRT with just one RT enveloped by the CP. Can we observe an RT *A* from the CP? The same concept applies to multiple RTs enveloped by the CP.

The answer is yes if Assumption 4 of Section II holds. The solution requires partitioning the RT into disjoint SRs such that each SR has only one adjacent disk as shown in Fig. 10(a) with vertices pattern described by Fig. 10(b). Fig. 10(a) shows LosPoRT regions performed by LosPoRT 1. There are three variants of the LosPoRT algorithms: LosPoRT 1–3. Table VI shows LosPoRT 1 algorithm. LosPoRT 1 partitions an RT

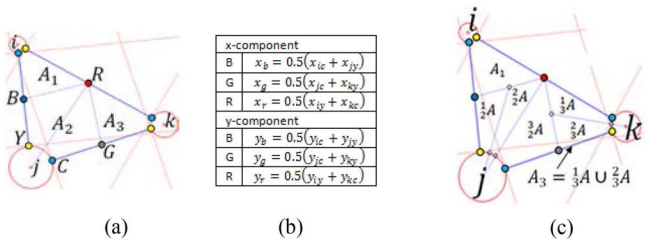


Fig. 10. LosPoRT and its vertices' relation. (x_{iy}, y_{iy}) and (x_{jy}, y_{jy}) are the coordinates of yellow EPs on disks i and j, respectively. (a) LosPoRT. (b) Points' geometry. (c) A_2's partition.

TABLE VI
LOSPORT ALGORITHM 1

Input: RT of 3 disks (e.g., disks $i, j,$ and k).
Output: SRs $A_1, A_2,$ and A_3. Each SR has only one adjacent disk.
Algorithm Steps Description:
1. Set w, o, and q as disks. Put disks i, j, k in the list, list LD.
2. Determine the longest of three edges of the RT. Compute the midpoint of the longest edge and name it as point R.
3. Name the disk opposite to R as disk o. Remove disk $i, j,$ or k that is equal to disk o from LD. Set $q = min(LD)$. Set $w = max(LD)$.
4. Compute the midpoints of the other two edges. Name the midpoints as G and B so that CW order of B, R, and G are maintained for SR adjacent to disk o.
5. Connect points B, R, and G to form $\overline{BR}$ and $\overline{GR}$.
6. Find A_1 which is adjacent to disk q and it is bounded by $CS(C_q, Y_q)$, $\overline{Y_qR}$, $\overline{RB}$, and $\overline{BC_q}$.
7. Find A_2 which is adjacent to A_1 and bounded by $CS(C_o, Y_o)$, $\overline{Y_oB}$, $\overline{BR}$, $\overline{RG}$, and $\overline{GC_o}$.
8. Find A_3 which is adjacent to A_2 and bounded by $CS(C_w, Y_w)$, $\overline{Y_wG}$, $\overline{GR}$, and $\overline{RC_w}$.

into three disjoint SRs: $A_1, A_2,$ and A_3. A_1 and A_3 have four vertices each and A_2 has five vertices. Points B, G, and R are unique within the RT. Points C and Y are not unique.

For this reason, points C and Y for disks i, j, and k are denoted as $C_i, Y_i, C_j, Y_j, C_k,$ and Y_k, respectively, if more than one presented in the same image. The coordinates for points, B, G, R, $C_i, Y_i, C_j, Y_j, C_k,$ and Y_k are defined as (x_b, y_b), (x_g, y_g), (x_r, y_r), (x_{ic}, y_{ic}), (x_{iy}, y_{iy}), (x_{jc}, y_{jc}), (x_{jy}, y_{jy}), (x_{kc}, y_{kc}), and (x_{ky}, y_{ky}).

In Table I, B, C, G, R, and Y represent blue, cyan, gray, red, and yellow, respectively. Fig. 10(c) shows how further partition of SR A_2 would look likes when it is partitioned by LosPoRT 2 which will be discussed in the next section. LosPoRT is motivated by the fact that the intersection of two sets is at most the size of the smaller of the two sets.

Before presenting Lemma 2, let us define a few definitions that will be needed throughout the rest of this paper. An RT may also be called RT 7 for example if it is the seventh RT in the TR. A CP that surrounds an RT A which is formed by disks i, j, and k may also be referred to as $P(A)$ or $P(i, j, k)$. A CP that surrounds RTs A and B may be referred to as $P(A, B)$.

A VCS of a region A with endpoints P_1 and P_2 is denoted as VCS(A) or VCS(P_1, P_2). An observer on VCS(A) can observe all points in region A. Each VCS, if only a segment of $P(A)$, is an open ended CS because its end points are not included. This is similar to the concept of open set. When we refer to a blind curve segment (BCS) we also include both end points of the CS. For example, a point on the BCS(A) cannot see A. Now we are ready to present Lemma 2.

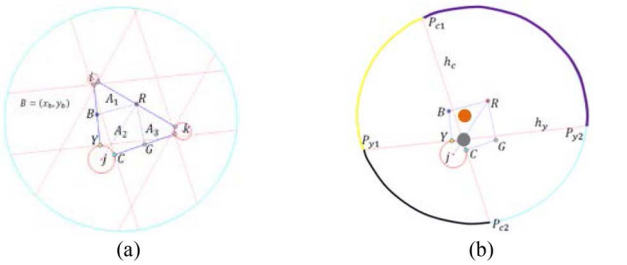


Fig. 11. LosPoRT region and its typical CP $P(i, j, k)$. (a) Regions A_1, A_2, and A_3. (b) LosPoRT region A_2.

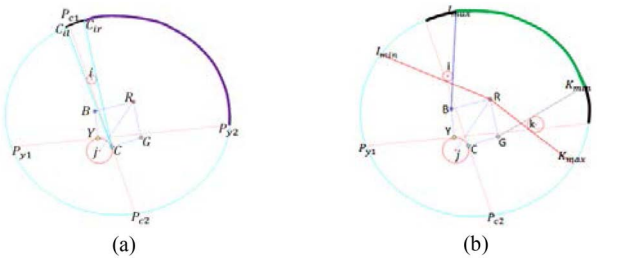


Fig. 12. Highlight the visibility of A_2's vertices. Using the concept of visible cone which originates at C, Y, B, R, or G then ends at two points on the CP. VCS is the CVCS with BCS removed. (a) Inclusion of disk i shows a BCS with respect to the vertex C only. (b) VCS when all points and all three disks are considered.

Lemma 2: The subset A_2, $A_2 \subset A = \cup_{l=1}^{3} A_l$, bounded by CW CSs, $\overline{BR}$, $\overline{GR}$, $\overline{GC_j}$, CS(C_j, Y_j), and $\overline{Y_jB}$, as shown in Fig. 11(a) can be observed from $P(i, j, k)$ with a single observer if there exists at least one VCS on $P(i, j, k)$ where an observer can be selected to see all vertices of A_2.

Before proving Lemma 2 let us observe one of the LosPoRT regions, A_2, as shown in Fig. 11(b). Any observer on the yellow CS and the purple CS can observe the Y vertex. Likewise any observer on the cyan CS and purple CS can observe the C vertex. Any observer on the black CS can neither observe Y nor C vertex. Any observer on the purple CS can observe Y and C vertex. The purple CS is the candidate visible curve segment (CVCS) because it is generated by the two vertices that are on the disk. In fact, any observer point in a region T, e.g., orange point, bounded by the CVCS, $\overline{P_{y2}G}$, and $\overline{GP_{c1}}$ can observe A_2 if there is no disk in T. G is the gray point shown in Fig. 11(b). Fig. 12(a) shows how a CVCS is blocked by disk i with respect to vertex C. Repeating the same pattern for all other vertices with disks i and k, the final VCS is shown in Fig. 12(b) in green color, VCS($I_{\max}, K_{\min}$). Now we are ready to prove Lemma 2.

Proof: Any region partitioned by the LosPoRT algorithm has at most five vertices. Since the region is part of an RT that is formed by three disks, there exist at least three potentially VCSs on $P(i, j, k)$, corresponding to the number of disks, from which an observer can be chosen that can observe the whole set of A_l where $l = 1, 2,$ and 3. The number of VCSs can also be directly obtained from drawing a visible cone from all vertices of subset A_l to enclose each of the disks inside the CP. The rays that originate from each vertex are tangent to each side of each disk. Any point on the VCS can observe A_l. ■

TABLE VII
LOSPORT ALGORITHM 2

Input: SRs $A_1, A_2,$ and A_3 partitioned by LosPoRT algorithm.
Output: SRs ${}_i^M A$ where $i \in 1,2,3$ and M is a user-specified number, the number of partition of SR $A_1, A_2,$ and A_3. Each SR is a LosPoRT SR.
Algorithm Steps Description:

1. If a region to be partitioned is A_2, and M = 2, then partition A_2 by drawing a line from the center of A_2's adjacent disk to the R vertex.
2. Else if a region to be partitioned is a SR of A_2, then partition the SR by drawing a line from the center of the A_2's adjacent disk to the midpoint of the edge that is opposite to the adjacent disk.
3. Else if a region to be partitioned is A_1, A_3, A_1's or A_3's SRs then partition the SR by drawing a line from the center of A_1's or A_3's adjacent disk to the midpoint of the edge that is opposite to the adjacent disk.

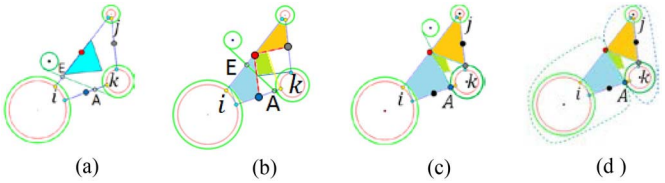


Fig. 13. Different scenarios of $\overline{AE}$ leading to LosPoRT 3 algorithm. (a) RT, VP, and $\overline{AE}$. (b) $\overline{AE}$ crossing A_1 and A_2. (c) $\overline{AE}$ crossing A_1, A_2, and A_3. (d) CPs crossing A_1, A_2, or A_3.

E. Auxiliary Algorithm 5 (LosPoRT 2 Algorithm)

LosPoRT 2 algorithm in Table VII is developed to observe an RT that the CP does not cross. It is basically the same as LosPoRT 1, except that it partitions SRs $A_1, A_2,$ and A_3 of LosPoRT 1 to obtain smaller SRs while potentially obtaining larger and more VCSs. LosPoRT 2 creates pattern described by Fig. 10(c).

If region A_i, $i \in 1, 2, 3$, is partitioned by LosPoRT 2, the mutually disjoined SRs of A_i maybe notated as ${}_i^M A$, where M is an integer representing the user specified number of partitioning. If M is equal to 2, then $A_i = {}_i^1 A \cup {}_i^2 A$.

F. Auxiliary Algorithm 6 (LosPoRT 3 Algorithm)

LosPoRT 3 algorithm is developed to partition an RT into three SRs. It is very similar to LosPoRT 1 because all three SRs are maintaining the same vertices. The differences are points G and B are moved closer to the disk adjacent to SR A_2. LosPoRT 3 is applied on the crossing RT that cannot be observed by the VP algorithm or the CSIRT algorithm as will be seen in steps 5 and 13 of the GreenPath and the BluePath algorithms, respectively. It is used by the GreenPath and the BluePath algorithms. Fig. 13 illustrates different combinations of RT and CS pairs. Fig. 13(a) shows that $\overline{AE}$ is extended from VC 2. $\overline{AE}$ is crossing RT 4, but does not cross VP 4. CSIRT algorithm does not work since there is no VD. The scenario requiring LosPoRT 3 is not limited to just CP's CS extending from a VC. More details will be discussed in Section IV.

If VC 2 moves slightly up so that $\overline{AE}$ marginally intersects VP 4, there is no guaranteed that RT 4 remains observable when the GreenPath undergoes modification by the BluePath algorithm to remove discontinuity. Fig. 13(b) suggests that LosPoRT 1 may partition the RT into SRs that can be observed from $\overline{AE}$; however, it is better to have a small A_2 because it

TABLE VIII
LOSPORT ALGORITHM 3

Input: The crossing RT with disks i, j, k, and the BluePath's CS.
Output: SRs A_1, A_2, and A_3. VPs $VP1, VP2$, and $VP3$. VCSs $VCS1$, $VCS2$, and $VCS3$. States $ST1$, $ST2$, and $ST3$. If all 3 states are true, then the input RT is observable from the input CS.
Algorithm Steps Description:

1. Set w, o, and q as disks. Put disks i, j, k in the list, list LD.
2. Determine the longest of three edges of the RT. Compute the midpoint of the longest edge and name it as point R.
3. Name the disk opposite to R as disk o. Remove disk i, j, or k that is equal to disk o from LD. Set $q = min(LD)$. Set $w = max(LD)$.
4. Enlarge disk o with the robot's radius, $R_o = R_o + R_r$.
5. Compute the intersections of disk o with the other two edges of the RT that are connecting to disk o, then name them as points G and B so that CW order of B, R, and G are maintained for SR adjacent to disk o.
6. Connect points B, R, and G to form $\overline{BR}$ and $\overline{GR}$.
7. Find A_1 so that it is adjacent to disk q and it is bounded by $CS(C_q, Y_q)$, $\overline{Y_qR}$, $\overline{RB}$, and $\overline{BC_q}$.
8. Find A_2 so that it is adjacent to A_1 and is bounded by $CS(C_o, Y_o)$, $\overline{Y_oB}$, $\overline{BR}$, $\overline{RG}$, and $\overline{GC_o}$.
9. Find A_3 so that it is adjacent to A_2 and is bounded by $CS(C_w, Y_w)$, $\overline{Y_wG}$, $\overline{GR}$, and $\overline{RC_w}$.
10. Compute the VP of A_1, A_2, and A_3 and name them as $VP1$, $VP2$, and $VP3$ respectively.
11. Set $ST1$, $ST2$, and $ST3$ to false. Set $VCS1$, $VCS2$, and $VCS3$ to 0.
12. Compute $\overline{AE}$. If $\overline{AE}$ crosses A_1 and $VP1$ set $ST1$ to true. If $\overline{AE}$ crosses A_2 and $VP2$ set $ST2$ to true. If $\overline{AE}$ crosses A_3 and $VP3$ set $ST3$ to true.
13. Compute $VCS1$, $VCS2$, and $VCS3$ with $\overline{AE}$ as the CP's CS.
14. If $VCS1$ is greater than 0, set $ST1$ to true. If $VCS2$ is greater than 0, set $ST2$ to true. If $VCS3$ is greater than 0, set $ST3$ to true.
15. Return true if $ST1$ and $ST2$ and $ST3$ are true. Else return false.

may be a noncrossing SR of the crossing RT. Fig. 13(c) suggests a better partition of the RT since $\overline{AE}$ crosses all three SRs. In addition, $\overline{AE}$ crosses the VPs of all three SRs shown in blue, green, and orange. In comparison, $\overline{AE}$ of Fig. 13(c) does not cross A_1 of Fig. 13(b).

The objective of LosPoRT 3 is to partition a crossing RT into three SRs with SRs A_1 and A_3 containing all of collision free segments of two edges of the RT. Fig. 13(c) shows that A_1 contains all collision free segments of $\overline{C_iY_k}$ and A_3 contains all collision free segments of $\overline{C_kY_j}$. The black points in Fig. 13(c) denote the blue and the gray points found by LosPoRT 1. LosPoRT 3 algorithm is shown in Table VIII. The algorithm used both VP and VCS concepts to determine observability. The CP does not have to completely surround the SR to have a VCS. Provided that the segment of the CP is in the CVCS of the SR, at least in the local region, and it is not blocked by any other disks, then there exist a VCS or a point that can observe A_1, A_2, or A_3. In Fig. 13(c), $|\text{VCS}(A_1)| = |\overline{AE}| \geq |\text{VCS}(A_3)|$. Note that this result is only for this RT and CP's CS pair within this RT. Global result for $\text{VCS}(A_1)$ and $\text{VCS}(A_3)$ may not be the same. The GreenPath's and the BluePath's algorithms only consider the CP's CS within the RT to maintain computational tractability. Result of step 13 is found by computing the CVCS of each LosPoRT 3 SR which is detailed in Section III-C. The SR A_2 in Fig. 12(b) can be observed from $\text{VCS}(I_{\max}, K_{\min})$ which is a segment of $P(i, j, k)$. Likewise, the result of Lemma 2 can be used with the RT and CS pair implemented with LosPoRT 3 by considering the CS as a segment of imaginary $P(A_l)$, $l \in 1, 2, 3$, which has no real physical presence other than $\overline{AE}$. Most of the CS of $P(A_l)$ is makeup to simulate scenario in Fig. 11. Fig. 13(d) shows that $|\text{VCS}(A_1)| = |\overline{AE}|$ and A_2 and A_3 are observable from $\overline{AE}$, $\overline{AE}$ is a segment of $P(A_l)$. $P(A_1)$ is the dashed green CP. $P(A_3) = P(A_2)$ is the dashed blue CP. $P(A_l)$ in Fig. 13(d) is different from $P(A_l)$ in Fig. 11(b), but the result of Lemma 2 still holds because $P(A_l)$ is enveloping both vertices C_l and Y_l of disk l. The CVCS of disk l will intersect $P(A_l)$ and then create a bounded region T which ensures observability as discussed in Section III-D with an orange point observer.

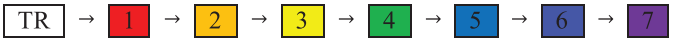


Fig. 14. Seven phases of the rainbow algorithm in rainbow color.

G. Auxiliary Algorithm 7 (Observer Placement Algorithm)

An observer placement (OP) algorithm is developed to manage data structures for different type of observer lists such as VCS observers, CSIRT observers, VCPM observers, and other observers for the TR. OP algorithm is a subalgorithm for the VioletPath algorithm. Its details can be found in Table XX. CSIRT observers and VCPM observers are selected to observe the TR first due to their in-flexibility. Observers on VCSs, VP observers and LosPoRT observers, are followed due to their flexibilities. OP algorithm finds sufficient number of observers to observe the TR.

IV. RAINBOW COVERAGE PATH PLANNING ALGORITHM

In this section, we discuss the rainbow algorithms presented in Table II. The rainbow algorithm is developed to find static or fixed CP for the robot to follow and also to find observers needed to sense the entire TR. Observers are constrained to be on the CP. Avoiding moving obstacles can cause the robot to deviate from the fixed CP and it will be addressed in Section V. Fig. 14 illustrates the sequence of phases in the rainbow algorithm. The rainbow algorithm begins by transforming the TR as an input to get the convex hull of the interior disks. The final outputs of the transformation are the static CP and all observers for the TR. The rainbow algorithm generates a closed CP that is a collection of LS and CS.

Fig. 15(a)–(g) shows the output CPs for all seven phases of the rainbow algorithm for the sample TR shown in Fig. 1. The number of vertices, LSs, and CSs increase from the RedPath to the GreenPath and then remain constant from the GreenPath onward as shown in Table X. The leftmost column in Table X indicates the phase of the rainbow algorithm. Common notations used to compute the rainbow algorithm is tabulated in Table IX. Fig. 15(a) and (b) do not illustrate the exterior disks of the TR because they are removed by step 1 of the RedPath algorithm and they are not needed by the OrangePath algorithm. The exterior disks of the TR are needed in the YellowPath algorithm and all other rainbow algorithms.

The polygons in cyan color illustrate in Fig. 15(d)–(f) are the VPs of the RTs. Fig. 15(h) shows the output CPs for all seven phases of the rainbow algorithm.

A. Foundation Path, the RedPath

The RedPath algorithm is developed to find a foundational path. The RedPath for the sample TR is shown in Fig. 16(a).

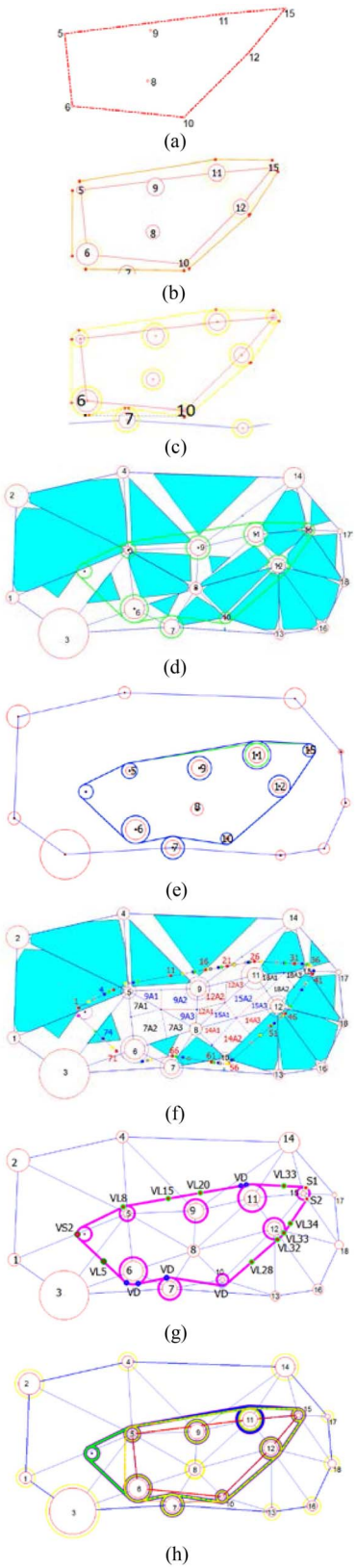


Fig. 15. CPs from phases 1–7 for the sample TR. (a) RedPath. (b) OrangePath. (c) YellowPath. (d) GreenPath. (e) BluePath. (f) IndigoPath. (g) VioletPath. (h) CPs (a)–(g).

All LSs that form the RedPath are described in Fig. 16(b). The RedPath algorithm is shown in Table XI. RedPath's third step implements existing Graham scan algorithm to find the convex hull. Hull vertices, points 5, 6, 10, 11, 12, and 15, are points where two LSs of the hull meet.

TABLE IX
SOME NOTATIONS USED IN THE RAINBOW ALGORITHM

N is the number of disks in TR.
I is the number of intersection for a set S of n LSs [24].
E is the number of edges according to Euler's formula.
xPLS is RPath's, OPath's, YPath's LS if x is R, O, or Y respectively.
N_i is the number of interior disks, $N_h \leq N_i < N$.
N_h is the number of interior disks with centers on the convex hull.
N_{col} is the number of disks involved in collision with OPLS.
N_{VCPM} is the number number of VC.
N_{cs} is the number of CS in the closed path.
N_{xcs} is the number of CS in the green CP or the BluePath if x is g or b respectively.
$N_h + N_{col} \leq N$.
M is the number of RTs, $(M + N - E = 1)$.
M = $(M_{VP} + M_{CSIRT} + M_{VCPM} + M_{LosPoRT})$.
M_{SR} is the number of SRs, $M_{SR} = 3M_{LosPoRT}$.

TABLE X
NUMBER OF RAINBOW ALGORITHM'S VERTICES, LSS, AND CSS

	Number of vertices	Number of LS	Number of CS
1	N_h	N_h	0
2	$2N_h$	N_h	N_h
3	$2N_h + 2N_{col}$	$N_h + N_{col}$	$N_h + N_{col}$
4	$2N_h + 2N_{col} + 2N_{VCPM}$	$N_h + N_{col} + N_{VCPM}$	$N_h + N_{col} + N_{VCPM}$

TABLE XI
REDPATH ALGORITHM

Input: The TR with all known static disks.
Output: A convex hull of interior disks' center.
Algorithm Steps Description:
1. Remove all exterior disks of the TR.
2. Reduce all remaining interior disks to points, interior points.
3. Find convex hull of interior points with Graham Scan algorithm.

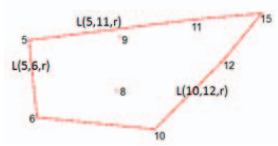
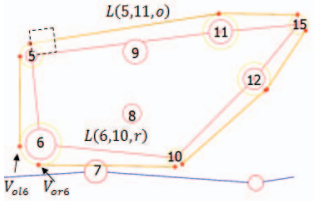
	RedPath's LSs in CW direction: L(5,11,r), L(11, 15, r), L(15, 12,r), L(12, 10,r), L(10, 6,r), and L(6, 5,r).
(a)	(b)

Fig. 16. RedPath. $L(i,\ j,\ r)$ or $L(V_i, V_j, r)$ is a red LS connecting points i and j or points V_i and V_j. The same notation is used for other colors by replacing r with y for yellow, etc. (a) Convex hull of interior disks' centers. (b) RedPath's LSs.

B. A Coverage Path Enveloping Hull Disks, OrangePath

An OrangePath is developed to find a path that envelopes all hull disks. To get the OrangePath from the RedPath, relative relationship of all LSs of the RedPath must be known and they can be determined by the BURL algorithm. The OrangePath algorithm, shown in Table XII, establishes the relationship between any connected RedPath's vertices, i and j, as $L(i,\ j,\ r)$ with the following algebraic equations:

$$\begin{aligned} y(i,j,r) = y(j,i,r) &= m(i,j,r)x + b(i,j,r), \text{ where} \\ m(i,j,r) = m(j,i,r) &= (y_i - y_j/x_i - x_j), \text{ and} \\ b(i,j,r) &= y_i - m(i,j,r)x_i. \end{aligned} \tag{3}$$



OrangePath's LSs in CW order: $L(V_{or5}, V_{ol11})$, $L(V_{or11}, V_{ol15})$, $L(V_{or15}, V_{or12})$, $L(V_{ol12}, V_{or10})$, $L(V_{ol10}, V_{or6})$, and $L(V_{ol6}, V_{ol5})$.
OrangePath's CSs in CW order: $CS(V_{ol11}, V_{or11})$, $CS(V_{ol15}, V_{or15})$, …, $CS(V_{or6}, V_{ol6})$, and $CS(V_{ol5}, V_{or5})$.

(a) (b)

Fig. 17. OrangePath enveloping convex hull's disks. This figure shows a CS, $CS(V_{or6}, V_{ol6})$, is connecting two LSs. The subscript o, r, and 6 represent orange color, right, and sixth vertex, respectively. Similarly, the subscript o, l, and 6 represent orange color, left, and sixth vertex. This representation is used throughout this paper for other colors in the rainbow algorithm. (a) OrangePath of the TR. (b) CW directions of LSs/CSs.

TABLE XII
ORANGEPATH ALGORITHM. THE SECOND RAINBOW ALGORITHM

Input: A convex hull of interior disks' center point (RedPath).
Output: A set of CSs enveloping the hull disks completely (OrangePath).
Algorithm Steps Description:
1. Determine the RedPath's LS status with the BURL algorithm.
2. Enlarged all points of the input to their original radius plus the robot's radius.
3. Compute CET LS to a pair of enlarged disks WRT equation (4), the status of each RPath's LS, and the end points of each LS.
4. Apply the index set of the input LS to the output LS.
5. Compute CS between any two vertices on the same enlarged disk as computed in step 2.

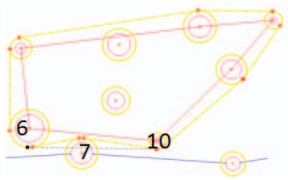
To find the OrangePath, all points that are on the RedPath must be enlarged by the robot's radius and the original radius of the point considered at a minimum to prevent collision with the disk when the robot moves along the path. Equation (3) and the BURL algorithm provide a solution to (4) which connects OrangePath's vertices, i and j, with $L(i,\ j,\ o)$. All OrangePath LSs are CET to hull disks

$$y(i,j,o) = m(i,j,o)x + b(i,j,o), \text{ where}$$
$$m(i,j,o) = m(j,i,o) \text{ and } b(i,j,o) = y_i - m(i,j,o)x_i. \quad (4)$$

$m(i,j,o) = m(j,i,o)$ is the slope between two endpoints of a LS of the OrangePath. Since disks i and j may have different radii, slopes $m(i,j,r)$ and $m(i,j,o)$ of (3) and (4) are not related. Note that the OrangePath also has CSs not just LSs. Slope of the CS of the OrangePath are simply the derivative of each of the CS' equation. Fig. 17(a) shows that any hull's disk has two different vertices generating a CS that connects two different LSs. For example, a hull disk 6, has vertices V_{or6} and V_{ol6} which form $CS(V_{or6},\ V_{ol6})$. The OrangePath may have collision with other interior or exterior disks in the TR. For example, $L(V_{ol10}, V_{or6})$ is colliding with disk 7. An OrangePath must be processed by the YellowPath algorithm to advance to the next phase.

C. Collision Free Path, YellowPath

The YellowPath is developed to find a path that envelopes all interior disks while avoiding collision with all disks in the TR. If the OrangePath has no collision with any disks in the TR, then an OrangePath and the YellowPath would be the same. The YellowPath algorithm only modifies the



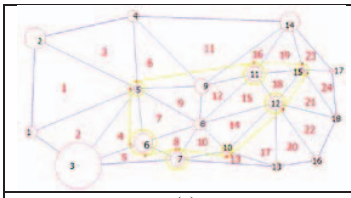
YellowPath's LS in CW order: $L(V_{yr5}, V_{yl11})$, $L(V_{yr11}, V_{yl15})$, $L(V_{yr15}, V_{yr12})$, $L(V_{yl12}, V_{yr10})$, $L(V_{yl10}, V_{yr7})$, $L(V_{yl7}, V_{yr6})$, and $L(V_{yl6}, V_{yl5})$.
YellowPath's CS in CW order: $CS(V_{yl11}, V_{yr11})$,…, $CS(V_{yr7}, V_{yl7})$, …, and $CS(V_{ol5}, V_{or5})$.

(a) (b)

Fig. 18. YellowPath. This shows a CIT collision correction. (a) YellowPath of the TR. (b) CW directions of LSs/CSs.

TABLE XIII
YELLOWPATH ALGORITHM. THE THIRD RAINBOW ALGORITHM

Input: OrangePath and all disks in the TR.
Output: The YellowPath.
Algorithm Steps Description:
1. Check if any input LS intersects any enlarged disk. If a LS is not intersecting any enlarged disk, promote the input LS to yellow LS. Else find a pair of LSs to replace the LS that causes collision based on the input LS and the enlarged disk involved in collision. Ensure the new LSs are CIT/CET to the two enlarged disks that they are connecting.
2. Update all vertices and CS data structure of the YellowPath.



YellowPath crosses the following RTs:
1-6, 8, 10-11, 13, 16, 17, 19-23, and 24. Crossing RTs decide how the YellowPath will be modified.

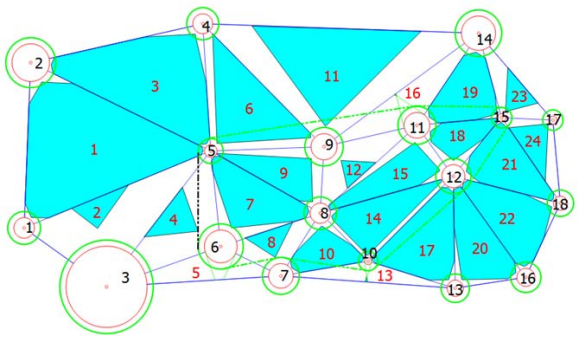
(a) (b)

Fig. 19. Transition from the YellowPath to the GreenPath with DT and identification of RTs that the YellowPath is crossing, crossing RTs. (a) YellowPath in the partitioned TR. (b) List of crossing RTs.

OrangePath's CS that have collision with a disk. Fig. 18 illustrates how a YellowPath algorithm modifies $L(V_{ol10}, V_{or6})$ of the OrangePath into $L(V_{yl10}, V_{yr7})$, $CS(V_{yr7},\ V_{yl7})$, and $L(V_{yl7}, V_{yr6})$. One collision results in one LS being replaced by two new LSs and a CS. Each time an LS is detected to collide with a disk, either CIT or CET collision correction is required depending on the status of the disk involved in collision. Because disk 7 is an exterior disk, CIT collision avoidance is applied to constraint the CP in the TR. An example of CET collision correction would occur if interior disk 9 is large enough that it intersects with $L(V_{yr5}, V_{yl11})$. The YellowPath algorithm is shown in Table XIII.

D. Observable and Collision Free Path, GreenPath

The GreenPath algorithm is developed to modify the YellowPath as necessary to get a CP that can observe all crossing RTs of the TR. To know whether an RT is observable by the YellowPath, DT of the TR has to be computed. Fig. 19 shows the exact partition of the sample TR into 24 RTs based on DT. The YellowPath crosses RTs 1–6, 8, 10, 11, 13, 16, 17, and 19–24 while envelopes RTs 7, 9, 12, 14, 15, and 18. RTs enveloped by the YellowPath are known as the noncrossing RTs. Fig. 20 shows that RT 5 does not have a VP within its boundary while RTs 13 and 16 have very tiny VP. For crossing RTs that have their VPs crossed by the YellowPath, they are observable from the YellowPath.



(a)

YellowPath does not cross the VPs of crossing RTs 2, 5, 8, 13, and 16. These RTs may require CSIRT, VCPM, or LosPoRT 3 algorithms for observability.

(b)

Fig. 20. Transition from the YellowPath to the GreenPath with computation of the VP. The promotion and demotion of YellowPath's LSs to green and black helps to determine the GreenPath. (a) All RTs and their VPs. (b) CSIRT/VCPM RTs.

TABLE XIV
GREENPATH ALGORITHM

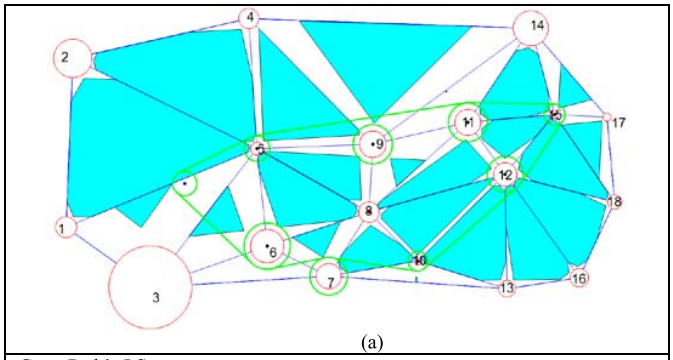
Input: The YellowPath, the TR, and all disks in the TR.
Output: A set of CSs forming a closed path. This new closed path can observe all RTs that it crosses (crossing RTs).
Algorithm Steps Description:

1. Perform DT[16] on the TR to find RT. Compute each RT's VP.
2. For each RT that the YellowPath crosses, add it to the crossingRT list.
3. For each crossing RT, check if the CS crosses it VP and if so promote the CS to green CS.
4. Else check if the RT and the CS can be partitioned by CSIRT algorithm and promote this CS to a green CS.
5. Else check if the RT and the CS can be partitioned by LosPoRT 3 and promote this CS to a green CS. Otherwise demote the CS to a black CS.
6. If the CS is a black CS, perform path modification through the VCPM algorithm and then pass the new CS(s) found to step 3. Else repeat step 3 with the new CS. If the current CS is the last CS in the list, end the program.

An and Qu [1] guaranteed that any observer inside of the VP of an RT can observe the RT. RTs 2, 5, 13, and 16 either have no VP or have VP not being crossed by the YellowPath. Visibility of RTs 5, 13, and 16 are solved with the CSIRT algorithm while visibility of RT 2 is solved by VCPM algorithm. In summary, visibility of crossing RT is either solved by VP [1], CSIRT, LosPoRT 3, or VCPM algorithms. The GreenPath for the sample TR is shown in Fig. 21. The GreenPath in Fig. 21 is longer than the YellowPath in Fig. 19 due to addition of two new LSs and a CS around RT 2 so that the new CP intersects VP 2. This is due to VCPM algorithm. The GreenPath algorithm is shown in Table XIV.

E. Differentiable, Observable, and Collision Free Path, BluePath

The BluePath is developed to modify the GreenPath as necessary to get a first order differentiable CP while maintaining observability of all crossing RTs. It is obvious from an OrangePath, a YellowPath, and a GreenPath that discontinuity exists in the CP. The discontinuity problem is treated in this section. The BluePath checks, and if necessary modifies, the CS to ensure differentiability. Differentiability correction occurs when $S_{gi} = (R_i + R_r)\theta_i = R_{gi}\theta_i$ is smaller than the user's specified threshold value (see Fig. 22). The BluePath



(a)

GreenPath's LSs:
$L(V_{gr5}, V_{gl11})$, …, $L(V_{gl7}, V_{gr6})$, and $L(V_{gl6}, C_{gl2})$, $L(C_{gr2}, V_{gl5})$.
GreenPath's CSs:
$CS(V_{gl11}, V_{gr11})$,…, $CS(C_{gl2}, V_{gr2})$, and $CS(V_{gl5}, V_{gr5})$.

(b)

Fig. 21. GreenPath which can observe all crossing RTs. (a) GreenPath of the sample TR. (b) GreenPath's CSs.

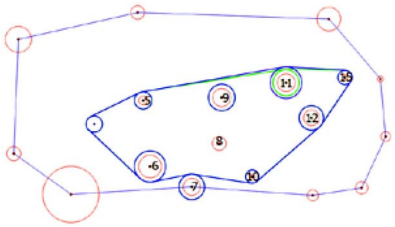


Fig. 22. Illustrating the enlargement of disk 11 to prevent discontinuity.

algorithm repairs differentiability problems by enlarging R_{gi} by the user specified value while keeping θ_i constant. θ_i is the central angle or the angle subtended at the center of a circle of radius R_{gi}. Every CS in a closed path is connected with two LSs. For example $CS(V_{gl11}, V_{gr11})$ or S_{g11} is connected with LSs $L(V_{gr5}, V_{gl11})$ and $L(V_{gr11}, V_{gl15})$. In addition, S_{g11} is a segment or arc of the enlarged disk 11. These descriptions are important in the BluePath algorithm.

Tables IX and X defined $N_{\text{gcs}} = N_h + N_{\text{col}} + N_{\text{VCPM}}$. N_{gcs} determines the running time of the BluePath algorithm. Fig. 22 shows an example of $N_{\text{gcs}} = 8$. The BluePath algorithm in Table XV determines that only S_{g11} has differentiability problem. Once S_{g11} is repaired, neighboring CSs S_{g5} and S_{g15} may be affected since they share $L(V_{gr5}, V_{gl11})$ and $L(V_{gr11}, V_{gl15})$, respectively. A state ST is required to generate the BluePath. ST(i) is true if S_{gi} is first order differentiable. Otherwise it is false. If S_{gi} is modified to obtain differentiability, then the two neighboring CSs, $S_{g(i-1)}$ and $S_{g(i+1)}$ needed to be checked and modified if necessary. In CW direction, if S_{gi} is S_{g11}, then $S_{g(i-1)}$ and $S_{g(i+1)}$ correspond to S_{g5} and S_{g15}, respectively. The example problem shown in Fig. 22 is a CET correction. If the discontinuity were to occur on disk 7, then a CIT correction would be required. The BluePath algorithm is shown in Table XV.

The output of the BluePath algorithm is a CP that is collision free, differentiable, and can observe all crossing RTs. The BluePath algorithm fails when Assumption 4 does not hold. The BluePath is input into the IndigoPath algorithm to obtain observability of noncrossing RTs. The noncrossing SR(s) of crossing RT(s) found by LosPoRT 3 may be inputted into

TABLE XV
BLUEPATH ALGORITHM. THE FIFTH RAINBOW ALGORITHM

Input: The GreenPath and all disks in the TR.
Output: A first order differentiable CP, BluePath, and crossing RTs' VCS and observers.
Algorithm Steps Description:
1. For $(i = 1, i < N_{gcs}, i++)$, set $ST(i)$ to false. Set temporary CSs $S_{ti} = S_{gi}$, $L(V_{tli}, V_{tr(i-1)})$ equal to $L(V_{gli}, V_{gr(i-1)})$, and $L(V_{tri}, V_{tl(i+1)})$ equal to $L(V_{gri}, V_{gl(i+1)})$.
2. For $(i = 1, i < N_{gcs}, i++)$, perform steps 3-6:
3. Check that S_{ti} satisfies the user's specified value. If so set $ST(i)$ to true then go to step 6. Else set $ST(i)$ to false.
4. If $ST(i)$ is false, set $ST(i-1)$ and $ST(i+1)$ to false.
5. Enlarge R_{gi} to find S_{ti} that satisfies the user's specified value. Compute and update $L(V_{tli}, V_{tr(i-1)})$ and $L(V_{tri}, V_{tl(i+1)})$. Set $ST(i)$ to true.
6. Set $i = i + 1$.
7. Repeat steps 2-6 until all $ST(i)$ are true and then promote S_{ti}, $L(V_{tli}, V_{tr(i-1)})$, and $L(V_{tri}, V_{tl(i+1)})$ to S_{bi}, $L(V_{bli}, V_{br(i-1)})$, and $L(V_{bri}, V_{bl(i+1)})$ respectively.
8. For $(j = 1, j < N_{gcs}, j++)$ perform the following steps:
9. If S_{bj} and S_{gj} are not equal, perform the following:
10. Set $CS_{bj} = merge\left(L(V_{blj}, V_{br(j-1)}),\ S_{bj}, L(V_{brj}, V_{bl(j+1)})\right)$. Check if CS_{bj} collide with any disk in the TR. If so, returns fail and exit the program. Else go to step 11.
11. Check if CS_{bj} cross all VPs of all crossing RTs. If so, go to step 14.
12. If any crossing RT's VP is not crossed by CS_{bj} in step 10, check if CS_{bj} and the RT pair can be partitioned by the CSIRT algorithm. If yes, go to step 14. Else go to step 13.
13. Check if CS_{bj} and the RT pair return true with LosPoRT 3. If yes, go to step 14. If not returns fail and exit the program.
14. Set $j = j + 1$.
15. For all crossing RTs, find their VCS(s), CSIRT, and VCPM observers.

TABLE XVI
SRS AND THEIR VCS(S) AS SHOWN IN FIG. 23

SR	VCS	SR	VCS	SR	VCS	SR	VCS
7A1	(71,73)	9A3	(76,5)	15A1	(31,36)	14A3	(60,68)
7A1	(57,67)	9A3	(7,12)	15A2	(59,64)	14A3	(3,4)
7A2	(9,15)	12A1	(21,26)	15A2	(50,56)	14A3	(15,21)
7A3	(74,6)	12A1	(33,37)	15A3	(14,20)	18A1	(30,36)
7A3	(8,11)	12A1	(47,51)	15A3	(75,1)	18A1	(41,45)
7A3	(66,70)	12A2	(48,56)	14A1	(29,36)	18A2	(27,35)
9A1	(10,17)	12A3	(52,56)	14A1	(58,61)	18A3	(24,31)
9A1	(62,68)	12A3	(58,63)	14A1	(47,54)	18A3	(43,46)
9A1	(46,53)	12A3	(14,22)	14A2	(16,23)	N/A	N/A
9A2	(72,2)	15A1	(47,54)	14A2	(28,32)	N/A	N/A
9A2	(65,69)	15A1	(18,25)	14A3	(49,56)	N/A	N/A

the IndigoPath algorithm to recompute VCS as larger VCS or more VCSs may exist. LosPoRT 3 and CSIRT algorithms are similar to each other and may complement each other. CSIRT algorithm remains useful, especially for RT 5' SR A_2. Since the CP is very close to disk 6, there is no VCS to observe A_2. When the robot's radius is much smaller, the CP is enveloping interior disks tighter than displayed in Fig. 21.

F. Differentiable, Observable, and Collision Free Path for the TR, IndigoPath

The IndigoPath is developed to find VCSs on the BluePath that can observe all noncrossing RTs and noncrossing SRs. It is implemented with LosPoRT 1–3. Fig. 9 illustrated a CP enveloping just one RT. Although the BluePath envelopes multiple RTs, the same concept of Section III-D is applied.

LosPoRT 1 finds SRs of an RT such that only one adjacent disk is in an SR. The SR is then used to find VCS(s) on the CP. LosPoRT 2 is not needed to solve the example

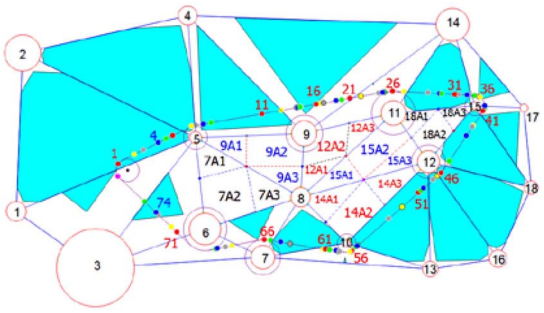


Fig. 23. LosPoRT of selected RTs and the VCSs it generated. The colored points on the IndigoPath are the endpoints of VCSs and they are considered in CW order. The VCSs displayed are only for RTs that the IndigoPath does not intersect. LosPoRT 3 is not used for the sample TR.

TABLE XVII
INDIGOPATH ALGORITHM. THE SIXTH RAINBOW ALGORITHM

Input: The BluePath, all noncrossing RTs, all noncrossing SRs, and all interior disks.
Output: Noncrossing SRs and their corresponding VCSs.
Algorithm Steps Description:
1. For each RT, perform LosPoRT 1 algorithm to obtain 3 SRs.
2. Find VCSs of each SR with all blocking interior disks.
3. If no VCS can be found, partition the SR into smaller SR with LosPoRT 2 algorithm and then repeat step 2 with the smaller SR.

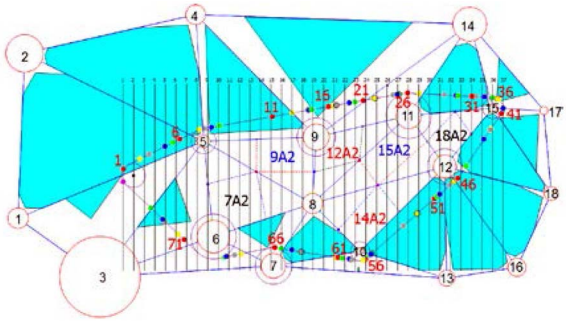


Fig. 24. VL test on the CP to find the intersection of VLs and VCSs.

problem. Fig. 23 and Table XVI show two VCSs where a point can be selected to observe SR $7A_1$: 1) CS(57,67) and 2) CS(71,73). The exact location of the observer on the VCS to observe a region is determined by the VioletPath algorithm. Table XVII shows an IndigoPath algorithm. For the sample TR, LosPoRT 3 never utilized.

G. Differentiable, Observable, and Collision Free Path for the TR With Observers, VioletPath

A VioletPath algorithm is developed to find a sufficient number of observers on the BluePath to observe the TR. This objective is achieved by finding a MER that enveloped all vertices of the BluePath. The dimension of the MER is then used to find equally spaced VLs as shown in Fig. 24. The number of VL required may be designed according to system requirement. The height of VL may be adjusted to exceed the height of the TR. The goal is to have sufficient number of VL such that most VCS intersect at least one VL. An intersection point of a VL and a VCS is a candidate observer point. Analogously, a HLT may also be employed

TABLE XVIII
INTERSECTION OF SELECTED VLT AND VCSS

	VL1	VL5	VL10	VL15	VL10	VL15	VL17
1	(72,2)	(74,6)	(7,12)	(7,12)	N/A	(57,67)	(57,67)
2	(74,6)	(76,5)	(8,11)	(8,11)	N/A	(60,68)	(60,68)
3	(76,5)	(3,4)	(9,14)	(9,14)	N/A	(62,68)	(62,68)
4	(75,1)	VP1	VP6	VP6	N/A	(66,69)	(59,64)
5	VP1	N/A	N/A	N/A	N/A	(66,70)	VP10

TABLE XIX
VIOLETPATH ALGORITHM. THE LAST RAINBOW ALGORITHM

Input: The BluePath, all crossing RTs, including crossing and noncrossing SRs, and their VCSs. CSIRT and VCPM observers. All SRs of the noncrossing RTs and their VCSs.
Output: Observers that jointly observe the TR.
Algorithm Steps Description:
1. Find a MER based on the input CP. Find the height of VL test.
2. Divide the width of a MER in step 1 by $(2N+1)$ to determine the spacing for $(2N+1)$ number of VL tests.
3. For each VL determine its intersection(s) with the VCSs along the "upper " CS of the BluePath. Store the intersection as observer point, VLUpt, and store all regions observed by VLUpt in VLUrg.
4. Repeat step 3 for the "lower" CS of the CP with VLBpt and VLBrg as observer point and region storage respectively.
5. Draw a different line test for each and every CSIRT observers. Store the intersection in CSIRTpt and store all regions observed in CSIRTrg.
6. Draw a different line test for each and every VCPM observers. Store the intersection in VCPMpt and store all regions observed in VCPMrg.
7. Call the OP algorithm with the results of steps 3, 4, 5, and 6 as inputs.

with a VL to generate a grid for intersection point computation. Fig. 24 shows VCSs and VLs intersection with some results tabulated in Table XVIII. The MER is divided vertically with (2*N*+1) number of VLs where N represents the number of disks in the TR. The interpretation of Table XVIII's column 2 is that VL1 is intersecting five VCSs which included the VCS generated by the VP of RT 1.

Results of OP algorithm are shown in Table XXII and Fig. 25. Notice that VL10 (VL10 underlined in column 6) does not intersect any VCS. This means that the point where VL10 intersects the BluePath is not a useful observer point.

Table XIX shows the VioletPath algorithm. Step 7 of the algorithm call OP algorithm to find observers for the TR. With the four assumptions in Section II, we summarize the results in Sections III and IV in Theorem 2.

Theorem 2: If the four assumptions in Section II hold, the rainbow CP planning algorithms will generate a CP with the following properties.

1) Collision-free in the presence of known disks.
2) First order differentiable.
3) Complete coverage of the TR.

Proof: In Section IV-A, we show that the RedPath is a convex hull of the interior points which are actually the center of the interior disks. The RedPath is a foundational path where the rest of the paths are improved upon. From the RedPath, we applied the BURL algorithm and algebraic manipulation to find the OrangePath. The OrangePath may or may not be collision-free and it is not completely visible. It is a foundational path where the YellowPath is derived from to find a collision free path. The YellowPath is collision free with all interior and exterior disks. From the YellowPath, the GreenPath is derived. The GreenPath inherits the collision free property from the YellowPath. In addition, the GreenPath is able to observe all RTs that intersect with the GreenPath. The GreenPath obtained visibility of all the RTs that it crosses through VP, CSIRT, LosPoRT 3, and VCPM algorithms. The GreenPath is then inputted to the BluePath algorithm to repair any differentiability problem. As a result, the BluePath is first order differentiable. The BluePath is then inputted to the IndigoPath algorithm which partitions the remaining RTs to which observability has not been verified. In addition the

TABLE XX
OP ALGORITHM

Input: VLUpt, VLUrg, VLBpt, VLBrg, CSIRTpt, CSIRTrg, VCPMpt, VCPMrg, and TargetRg, the TR with all RTs and their partitioned SRs.
Output: A collection of observers that observe the entire TR together.
Algorithm Steps Description:
1. Initialize all regions of the TR, e.g., RT1 and 7A3, to "not observed".
2. Merge VLUpt and VLBpt into VLpt. Sort VLpt based on the most number of regions in VLUrg and VLBrg that each VLpt can observe.
3. While removing CSIRTpt from the list one at a time, initialize TargetRg's region to "observed" based on CSIRTpt's CSIRTrg.
4. While removing VCPMpt from the list one at a time, initialize the TargetRg's region to "observed" based on VCPMpt's VCPMrg.
5. While removing VLpt from the list one at a time, initialize the TargetRg's region to "observed" based VLpt's regions in VLUrg and VLBrg.
6. If all observer points have been tested and there still exist "not observed" region, then place an observer where the CS of the BluePath first intersects the VCS of the "not observed" region in CW direction. Update the TargetRg's region as "observed". Else end this program.
7. Repeat step 5.

TABLE XXI
STATIC WORKSPACE CONFIGURATION IN ASSUMPTION 4 [1]

	O_{ix}	O_{iy}	R_i		O_{ix}	O_{iy}	R_i		O_{ix}	O_{iy}	R_i
1	1656	2927	22	7	2198	3028	26	13	2566	3054	17
2	1670	2579	40	8	2281	2896	21	14	2615	2517	36
3	1830	3051	85	9	2289	2756	27	15	2664	2695	9
4	2033	2497	21	10	2382	2997	7	16	2715	3031	19
5	2050	2765	13	11	2485	2711	27	17	2772	2700	8
6	2071	2966	35	12	2562	2818	24	18	2788	2875	15

TABLE XXII
ALL OPS AND THEIR COVERAGE REGIONS (RG)

	OP	RG	RG	RG	RG	RG	RG	RG
1	Vd11L	12A1	15A1	RT16	-	-	-	-
2	Vd11R	12A1	15A1		-	-	-	-
3	Vd10	RT13	-	-	-	-	-	-
5	Vd7R	7A1	7A3	9A1	9A2	14A3	RT8	-
6	Vd7L	7A1	7A3	9A1	9A2	14A3		-
7	Vd6	RT6	-	-	-	-	-	-
9	Vs2	7A3	9A2	9A3	15A3	RT2	RT1	-
10	VL8	RT3	-	-	-	-	-	-
11	VL15	7A2	RT6	7A3	9A1	9A3	-	-
12	VL20	RT11	12A3	15A3	14A3	-	-	-
13	VL33	RT19	14A2	18A3	14A1	-	-	-
14	VL5	RT4	9A2	-	-	-	-	-
15	VL28	12A3	15A1	14A3	15A2	14A1	RT17	12A2
16	VL32	RT20	9A1	12A1	12A2	15A1	14A1	-
17	VL33	RT22	18A3	-	-	-	-	-
18	VL34	RT21	-	-	-	-	-	-
19	S1	RT23	18A1	12A1	15A1	14A1	-	-
20	S2	RT24	12A1	15A1	18A1	-	-	-

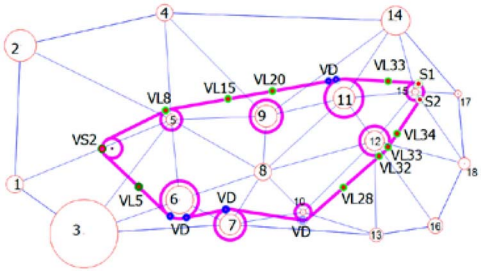


Fig. 25. Final rainbow CP with all observers.

IndigoPath algorithm computes the VCSs of the noncrossing RTs and noncrossing SRs. Finally the VioletPath inherits the collision free CP that is first order differentiable and has complete coverage of the TR after VLT algorithm computes the sufficient number of observers from the VCSs, VCPM observer, and CSIRT observers. ■

V. Simulation

The sample TR in Fig. 1 is tabulated in Table XXI. Table XXII illustrates the observers selected to observe the entire TR based on the rainbow algorithm. The order of the observers in the table is based on the OP algorithm in Table XX.

In Table XXII, Vd11L and Vd11R represent the observer to the left and to the right of VD 11, respectively. Vd10 represents the observers to RT 13. Vs2, VL8, *S*1, and *S*2 represent the observer at the VC 2, on the intersection of VL 8 with the BluePath, and on the intersection of special VL and the BluePath, respectively. Vd11L and Vd11R can observe regions 12A1 and 15A1 separately, but both must jointly observe RT 16. RTs 6 and 13 take two observers each to be observed. Note the underline observers VL28, VL32, VL33, and VL34 in Table XXII, the underlined simply means that the observer is on the lower CS of the CP. Observers represented by *S*1 and *S*2 simply mean that they are found by step 6 of the OP algorithm. The entry highlights in yellow represent the region already covered by preceding observer(s). The entry highlights in green means it takes two observers to observe one RT. Equivalently, the observers and their corresponding regions of coverage are shown in Fig. 25.

We simulate the data in Table XXI with CGAL [16] to obtain all the figures and results illustrated in this paper. All CSIRT observers are shown in blue with the word "VD" printed. Of particular interest is that RT 8 and RT 13 take two observers to be observed. In Fig. 25, they appear to be only 1 in RT 13, but they are two observers that are very close to each other. Some positive attributes of the rainbow algorithm include complete coverage through exact partition, collision free and first order differentiable CP. Some disadvantages of the rainbow algorithm include the VCPM algorithm which may produce a CS that is longer than necessary. CSIRT algorithm, which computes three observers on the CS within the RT, limit an ability to minimize the number of observers necessary to observe the entire TR.

A. Model of Differential Robot

A differential robot introduced in Section II is shown in Table XXIII with world coordinates represented by x and y. Robot's chained form coordinates are represented by z_1, z_2, and z_3.

TABLE XXIII
Motion Representation for Third Order System

Motion in world coordinate	Motion in chained form
$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix}=\begin{pmatrix}u_1 cos(\theta)\\ u_1 sin(\theta)\\ u_2\end{pmatrix}$	$\begin{pmatrix}\dot{z}_1\\ \dot{z}_2\\ \dot{z}_3\end{pmatrix}=\begin{pmatrix}v_1\\ v_2\\ z_2 v_1\end{pmatrix}$

TABLE XXIV
Coordinate and Input Transformation

Coordinate transformation	Input transformation
$\begin{pmatrix}z_1\\ z_2\\ z_3\end{pmatrix}=\begin{pmatrix}x\\ tan(\theta)\\ y\end{pmatrix}$	$\begin{pmatrix}u_1\\ u_2\end{pmatrix}=\begin{pmatrix}v_1 sec(\theta)\\ v_2 cos^2(\theta)\end{pmatrix}$

This sample model of the nonholonomic robot is included to show how useful the CP of the rainbow algorithm in the overall system performance. Based on the results of [15], our patrolling algorithm is as follows.

1) Select coordinates (x, y) and orientation of the robot such that $\theta \neq (\pi/2)$, apply state and input transformations as shown in Table XXIV and determine the corresponding boundary conditions $z^0 = [z_1^0,\ z_2^0, z_3^0],\ z^f = [z_1^f,\ z_2^f, z_3^f]$ to obtain the chained form motion or dynamic.
2) Let T_j be the time for the mobile robot to complete its maneuver between the adjacent pair of points and T_s^j be the sampling period such that $\overline{k} = (T_j/T_s^j)$ is an integer. The center of the moving disks O_i moving at constant velocities $v_i^k \triangleq [v_{i,x}^k v_{i,y}^k]$ for $t \in [t_0^j + kT_s^j,\ t_0^j + (k+1)T_s^j]$ are located at (x_i^k, y_i^k) during the time when $t = t_0 + kT_s$. For time $k = 0, \ldots, \overline{k} - 1$, recursively determine constants a_4^k from the following equations:

$$\min_{t\in[t_i^*, \bar{t}_i^*]} \quad g_2(z_1(t),k)\left(a_4^k\right)^2 + g_{1,i}(z_1(t),k,\tau)a_4^k + g_{0,i}(z_1(t),k,\tau)\ \geq 0 \tag{5}$$

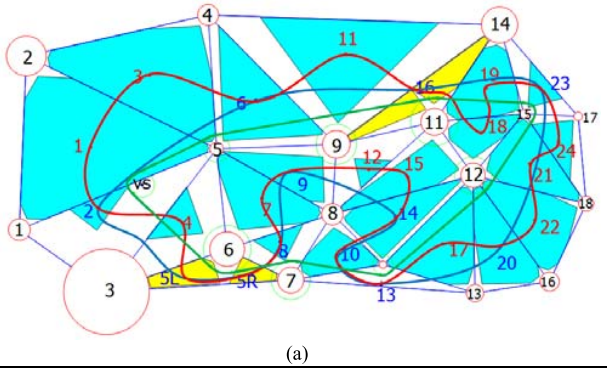
$$\tau = t - t_0 - kT_s \tag{6}$$

$$g_2(z_1(t),k) = \left[(z_1(t))^4 - \underline{f}(z_1(t))\left(B^k\right)^{-1}A^k\right]^2 \tag{7}$$

$$g_{1,i}(z_1(t),k,\tau) = 2\left[(z_1(t))^4 - \underline{f}(z_1(t))\left(B^k\right)^{-1}A^k\right]\left[\underline{f}(z_1(t))\left(B^k\right)^{-1}Y^k - y_i^k - v_{i,y}^k\tau\right] \tag{8}$$

$$g_{0,i}(z_1(t),k,\tau) = \left[\underline{f}(z_1(t))\left(B^k\right)^{-1}Y^k - y_i^k - v_{i,y}^k\tau\right]^2 + \left(z_1(t) - x_i^k - v_{i,x}^k\tau\right)^2 - (R_i + R_r)^2 \tag{9}$$

$$B^k = \begin{bmatrix} 1 & z_1^k & \left(z_1^k\right)^2 & \left(z_1^k\right)^3 \\ 0 & 1 & 2z_1^k & 3\left(z_1^k\right)^2 \\ 1 & z_1^f & \left(z_1^f\right)^2 & \left(z_1^f\right)^3 \\ 0 & 1 & 2z_1^f & 3\left(z_1^f\right)^2 \end{bmatrix} \tag{10}$$



(a)

Algorithm	Number of Observer	Length (m)
CWCP	26	3216
PNWCC	12	2390
Rainbow	20	2001

(b)

Fig. 26. Triangular-based CPs. Rainbow CP is the shortest of all three CPs. (a) Triangular-based CPs. (b) Triangular-based CPs and their performance when = 500.

$$Y^k = \begin{bmatrix} z_3^k \\ z_2^k \\ z_3^f \\ z_2^f \end{bmatrix}, A^k = \begin{bmatrix} {z_1^k}^4 \\ 4\left(z_1^k\right)^3 \\ {z_1^f}^4 \\ 4\left(z_1^f\right)^3 \end{bmatrix}$$

$$f(z_1) = \begin{bmatrix} 1 & z_1 & z_1^2 & z_1^3 \end{bmatrix}. \quad (11)$$

3) A feasible and collision-free path in the transformed state can be parameterized in polynomial/matrix form as

$$z_3(z_1) = F(z_1) = a^k f(z_1), \text{ where} \quad (12)$$

$$a^k = \left[a_0^k, a_1^k, a_2^k, a_3^k, a_4^k\right] \quad (13)$$

$$f(z_1) = \left[1,\ z_1(t), (z_1(t))^2, (z_1(t))^3, (z_1(t))^4\right]^T \quad (14)$$

$$\left[a^k\right]^T = \left(B^k\right)^{-1}\left(Y^k - A^k a_4^k\right). \quad (15)$$

4) The steering inputs to achieve path (12) which is a subset of $s(t)$ in (2) are determined to be

$$v_1(t) = C = \frac{z_1^f - z_1^0}{T}$$

$$v_2(t) = \left(2a_2^k + 6a_3^k z_1^k + 12a_4^k\left(z_1^k\right)^2\right)C + \left(6a_3^k + 24a_4^k z_1^k\right)(t - t_0 - kT_s)C^2 + 12a_4^k(t - t_0 - kT_s)^2C^3.$$

B. Path Comparison

Three CP planning based on exact partition algorithms are compared in Fig. 26. The red CP is generated by the CWCP algorithm [1]. It requires 26 observers and has a length of 3216 m. The blue CP is generated by previous next waypoint coverage constraint (PNWCC) algorithm [2]. It requires 12 observers and has a length of 2390 m. The green CP is generated by the rainbow algorithm. It requires 20 observers and has a length of 2001 m. Our rainbow algorithm reduces the CP length as well as the number of turns when compared to CWCP and PNWCC algorithms as illustrated in Fig. 26. The number of turns in the CP is proportional to the number of CSs. They are shown in Fig. 26(a).

TABLE XXV
RECENT COVERAGE TECHNIQUES

Ref.	Technique applied	Feature
[5]	(1) High resolution grid map representation. (2) Coarse to fine constrained inverse distance transform.	Path smoothing.
[6]	(1) Triangular-cell based map. (2) Distance transform based path planning.	Navigation flexibility.
[7]	(1) Neural network.	Low computation.
[12]	(1) Ulusoy. (2) Eulerian tour. (3) Chinese postman problem. (4) Rural postman problem. (5) Generalized Voronoi diagram. (6) Minimum perfect matching.	Energy constraint.

C. Performance Comparison

LosPoRT algorithms allow noncrossing RTs or SRs to be observed from a CP. Overlapping VCSs allow efficient OP along the CP. For example, one of the observers is placed at the intersection of the VL28 with the VioletPath because of eight intersections (see row 15 of Table XXII). In Section IV, we showed that our rainbow algorithm applied several well-known algorithms such as the Graham scan, DT, and line intersection. All mentioned algorithms can find our solution in $N\log(N)$ time.

Table XXV shows techniques and results of recent papers that achieved complete coverage in unknown TR. Lee *et al.* [5] considered circular and polygonal obstacles and its resulting path is smooth with few sharp turns. Oh *et al.* [6] and Luo *et al.* [7] considered polygonal obstacles and obtained zigzag CPs which are neither energy efficient nor time efficient.

Yazici *et al.* [12] considered rectangular obstacles and obtained zigzag CP. Although our obstacles are assumed circular, they can be relaxed to that of ellipsoid without affecting our rainbow algorithm provided that our CSIRT and LosPoRT algorithms generate coverage observers. In the case of polygonal obstacle, it can be enveloped by an ellipsoid which means that a differentiable CP may be kept, however, the visibility analysis has to be modified based on the art gallery theorem [17] or level set technique. Since our RedPath is a convex hull, our rainbow algorithm can be generalized to higher dimensions. In summary, our CP planning technique is better than the techniques listed in Table XXV because it provides complete coverage, smooth, short, segmentable, and curvature control CP, choice of OP, and fast running time as shown in Table XXVI.

D. Running Time and Convergence

A number of data structures are required to compute the running time in terms of big O notation. Most of the data structures are used in multiple phases of the rainbow algorithm. For example, the LineSegment class is used in all seven phases of the rainbow algorithm ranging from the RedPath to the VioletPath. Some member variables and function members

TABLE XXVI
RUN TIME OF THE RAINBOW ALGORITHM

	Selected activities	Running Time	Iteration
1	Graham Scan [23]	$N_i \log N_i$	-
2	BURL	$N_h \log N_h$	1
	Compute vertices	N_h	1
3	Collision correction	$N_h(N - N_h)$	3
4	Triangulation [16]	N log N	-
	Compute VP	M	1
	YPLS intersect RT	$(N_h + N_{col})M$	1
	YPLS intersect VP	$(N_h + N_{col})M$	1
	CSIRT	M_{CSIRT}	1
	LosPoRT 3	$M_{LosPoRT}$	
	VCPM	M_{VCPM}	3
5	Disk enlargement	$N_h + N_{col} + N_{vcpm}$	3
6	Partition, LosPoRT	$M_{LosPoRT}$	1
	Sub Region to CVCS mapping		
	CVCS	$(N_h + N_{col} + N_{VCPM})(M_{LosPoRT})$	1
	Find blocking disk(s) in the CVCS	N_i	1
	VCS for each SR	$(M_{SR})(N_i)$	1
7	Worst case line intersection for vertical grid with the followings		
	CSIRT Observer	$(M_{CSIRT} + N + I)log(M_{CSIRT} + N)$	1
	VCPM Observer	$(M_{VCPM} + N + I)log(M_{VCPM} + N)$	1
	VP observer	$(M_{VP} + N + I)log(M_{VP} + N)$	1
	LosPoRT's VCSs	$(M_{VCS} + N + I)log(M_{VCS} + N)$	1

are only used in some phases. For example a member function (MF) AddLineSegment() is only used in the YellowPath and the GreenPath. A MF ModifiedLineSegment() is used in an OrangePath, the YellowPath, the GreenPath, and the BluePath. Rainbow algorithm requires many other classes such as RT, VP, and SR to name a few.

Table XXVI shows how much time it takes to compute the most intensive activities in each of the color phase of the rainbow algorithm. The leftmost column represents by {1, 2, 3, 4, 5, 6, 7} are for the phase of the rainbow algorithm, respectively. Most activities are not converged as the number of input *N* approach infinity which implies that *M* also approach infinity. The slowest activity is in checking the intersection of the yellow path line segment with the RT and the VP of the GreenPath algorithm. Depending on how much modification is done in the BluePath algorithm, a complete repeating of the all activities in the GreenPath can be assumed as the upper bound running time for the BluePath. Running time in Table XXVI is not affected with this assumption.

A nested for loops was implemented which result in quadratic runtime. Table VI shows that the rainbow algorithm can compute the CP and observers for the TR with quadratic runtime. Locally, all of our algorithms do converge in at most three iterations where we compute the CET or CIT LS to correct collision, visibility, and first order differentiable problems. A simple algebraic manipulation and the BURL algorithm return the correct result of OPLS from the RPLS in one iteration.

VI. CONCLUSION

Our rainbow algorithm results in a complete and first order differentiable CP with static disks avoidance which means it can be implemented with time and energy efficiency. The dynamic obstacle avoidance technique can be incorporated as shown in Section V. Control points can be selected along the CSs that form the VioletPath. If dynamic obstacles are encountered, the dynamic obstacle avoidance algorithm can be executed by using two known points in world coordinate which are then transformed into chained coordinate and finally applied motion planning algorithm to avoid the obstacles. Once obstacle avoidance is performed, the robot can switch back to the static CP that has been planned. More specifically for our example problem, if a dynamic obstacle is blocking the robot from reaching an intended observation point VL15 on Fig. 25 as an example. As the robot moves from points VL8–VL15, the robot may not get to VL15 due to a moving obstacle. Depending on the criticality of the mission, the robot can slow down or avoid the moving obstacle at the cost of not getting to VL15. But the robot knows that VL15 observes a few regions, 7A2, RT6, 7A3, 9A1, and 9A3, as seen in Table XXII. By not getting to VL15, the robot can still search its databases to query which other observer points can help make up for the loss of visibility. For example observer Vd7L can cover regions 7A3 and 9A1, observer Vs2 can cover 9A3, and observer Vd6 can cover RT6. The concern for the robot at this point is to generate a new subpath that has observer that can observe region 7A2. We know that this is doable with the rainbow algorithm. Qu *et al.* [15] detailed how dynamic obstacle avoidance works and based on that work, we generate the dynamic part of the algorithm seen in Section V-A. In the case where complete coverage is not required, if other important parameters are also considered, a utility function can be imposed to weight and implement the decision. Integration under the curve can quickly find the area which consideration can be taken to cover a certain region of the TR or not based on the cost imposed on the distance of the path travel or the cost of computation on the sensor. A CP generated by our technique is shorter than a CP obtained in [1], [2], and [12] due to the foundation path obtained through the Graham scan algorithm and interior disks. Optimality condition in term of CP length may be obtained in its current form if there is no condition for VCPM. However, difficulty remains if both optimal conditions for path length and sensor placements are desired. Our approach can also be modified to adapt to unknown environments by applying the seven templates-based approach implemented in [6] and initial navigation through the known boundary of the TR. Last, but not least, the CP generated by the rainbow algorithm is very easy for the robot to follow because most of the segments of the CP are made of LSs and there are very few turns dictated by CSs. Recent advances in control of nonlinearities systems [25]–[28] make our contribution of the rainbow algorithm very promising in future works of path planning, surveillance, and other electronic applications.

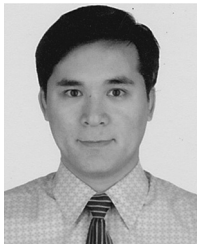
ACKNOWLEDGMENT

The authors would like to thank all the anonymous reviewers and the editors at SMC for meaningful comments.

REFERENCES

[1] V. An and Z. Qu, "Triangulation-based path planning for patrolling by a mobile robot," in *Proc. Aust. Control Conf.*, Perth, 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] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*. London, U.K.: Springer, 1998, pp. 203–209.
[4] H. Choset, "Coverage for robotics—A survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, nos. 1–4, pp. 113–126, 2001.
[5] T.-K. Lee, S.-H. Baek, S.-Y. Oh, and Y.-H. Choi, "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.
[6] 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.
[7] C. Luo, S. X. Yang, D. A. Stacey, and J. C. Jofriet, "A solution to vicinity problem of obstacles in complete coverage path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1. Washington, DC, USA, 2002, pp. 612–617.
[8] 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.
[9] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 718–724, Feb. 2004.
[10] C. Yang and K.-W. Chin, "Novel algorithms for complete targets coverage in energy harvesting wireless sensor networks," *IEEE Commun. Lett.*, vol. 18, no. 1, pp. 118–121, Jan. 2014.
[11] J. H. Lee, J. S. Choi, B. H. Lee, and K. W. Lee, "Complete coverage path planning for cleaning task using multiple robots," in *Proc. Int. Conf. Syst. Man Cybern.*, San Antonio, TX, USA, 2009, pp. 3618–3622.
[12] 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.
[13] W. Meiting, T. S. D. Junjian, and Y. Liwen, "Complete coverage path planning of wall-cleaning robot using visual sensor," in *Proc. Int. Conf. Electron. Meas. Instrum.*, Xi'an, China, 2007, pp. 4-159–4-164.
[14] 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.
[15] 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.
[16] A. Fabri, P. Alliez, and E. Fogel, "Computational geometry algorithms library," *Personal Comm*, 2004.
[17] J. O'Rourke, *Art Gallery Theorems and Algorithms*, vol. 57. New York, NY, USA: Oxford Univ. Press, 1987, pp. 1–29.
[18] J. Yang, Z. Qu, J. Wang, and K. Conrad, "Comparison of optimal solutions to real-time path planning for a mobile vehicle," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 4, pp. 721–731, Jul. 2010.
[19] Y. Guo and Z. Qu, "Coverage control for a mobile robot patrolling a dynamic and uncertain environment," in *Proc. 5th World Congr. Intell. Control Autom. (WCICA)*, vol. 6. Hangzhou, China, 2004, pp. 4899–4903.
[20] J. O'Rourke, "Finding minimal enclosing boxes," *Int. J. Comput. Inf. Sci.*, vol. 14, no. 3, pp. 183–199, 1985.
[21] D. A. Ross, *Master Math: Geometry*. Boston, MA, USA: Delmar Learn., 2004.
[22] K.-T. Leung and S. N. Suen. *Vectors, Matrices and Geometry*. vol. 1. Hong Kong: Hong Kong Univ. Press, 1994.
[23] R. L. Graham, "An efficient algorith for determining the convex hull of a finite planar set," *Inf. Process. Lett.*, vol. 1, no. 4, pp. 132–133, 1972.
[24] M. de Berg *et al.*, "Line segment intersection," in *Computational Geometry*. Berlin, Germany: Springer, 1997, pp. 19–43.
[25] 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.
[26] C. Chen, Z. Liu, Y. Zhang, and C. L. P. Chen, "Modeling and adaptive compensation of unknown multiple frequency vibrations for the stabilization and control of an active isolation system," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 3, pp. 900–911, May 2016.
[27] G. Lai, Z. Liu, Y. Zhang, C. L. P. Chen, and S. Xie, "Asymmetric actuator backlash compensation in quantized adaptive control of uncertain networked nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 2, pp. 294–307, Feb. 2017.
[28] 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.
[29] M. T. Goodrich, R. Tamassia, and D. Mount, *Data Structures and Algorithms in C++*, 2nd ed. Danvers, MA, USA: Wiley, 2009, pp. 594–654.
[30] J. Thomas, A. Blair, and N. Barnes, "Towards an efficient optimal trajectory planner for multiple mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 3. Las Vegas, NV, USA, 2003, pp. 2291–2296.



Vatana An (M'13) received the master's degree in electrical engineering from the University of Central Florida, Orlando, FL, USA, in 2005, where he is pursuing the Ph.D. degree.

He is with the U.S. Navy, Panama City, FL, USA. His current research interests include networked systems, cooperative controls, robotics, and autonomous vehicle systems.

Mr. An was featured in *High Tech* magazine in 2012 for his contribution to the U.S. Navy.

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, FL, USA, where he is currently a Pegasus Professor and the Chair of the Department of Electrical and Computer Engineering. His current research interests include nonlinear systems and control. He has published a number of papers in the above areas and authored three books entitled *Robust Control of Nonlinear Uncertain Systems* (Wiley Interscience, 1998), *Robust Tracking Control of Robotic Manipulators* (IEEE Press, 1996), and *Cooperative Control of Dynamical System* (Springer-Verlag, 2009).

Dr. Qu is currently an Associate Editor of *Automatica* and the *International Journal of Robotics and Automation*.

Rodney Roberts (M'91–SM'02) received the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1992.

He has been with Florida A&M University—Florida State University College of Engineering, Tallahassee, FL, USA, since 1994, where he is currently a Professor. His research interests include robotics and image processing.