Coverage Control for A Mobile Robot Patrolling A Dynamic and Uncertain Environment

Yi Guo and Zhihua Qu

***Abstract*— In mobile robot applications such as cleaning and security patrolling, a fundamentally important problem is to design feasible trajectories and steering control so that the robot moves collision-free and covers all the points (in its sensor/effector range) in a dynamic and uncertain environment. We formulate such a problem and propose constructive algorithms in sequential modules to solve it. First, a minimum-area rectangle is placed encasing the boundary of the set to be covered. Second, minimum number of circles of the radius of coverage range are placed to completely cover the rectangle. Third, a patrolling path is searched along the boundary of the set in a spital. Feasible trajectories are then designed to account for the nonholonomic kinematics of the robot and to avoid collisions from the dynamic obstacles detected by the robot onboard sensors. Since analytic solutions are given in generating feasible trajectories, the algorithm can be implemented in real time.**

I. Introduction

Motion planning for a mobile robot to reach a goal position from its start position has been intensively studied for decades. Up to date, limited research attention has been received to the problem of planning a path of complete coverage of an environment by a mobile robot, although such a problem is very important in various applications such as clearing, security patrolling, and sensor network deployment (see [1], [2], [3], [4]). Particularly, considering the nonholonomic kinematic constrains posed by the mobile robot, and the constrains posed by dynamic obstacles in the uncertain environment, the complete coverage path planning and control problem becomes challenging. Recent progress has been made in [5] to generate real time feasible trajectories using parameterized polynomial steering control for nonholonomic robots moving in 2D dynamically changing environments. This paper is to study the complete coverage path planning problem, and extend the results of [5] to the so-defined patrolling control problem.

In the remainder of the paper, we first formulate the patrolling control problem and give necessary assumptions in Section 2. And then in Section 3, patrolling control design is proposed with constructive algorithms given. Finally, the paper is concluded in Section 4.

II. Problem Formulation

We consider the problem of designing a steering control for a mobile robot to patrol a connected region in a dynamic and changing environment. As shown in figure 1, patrolling control is to generate a trajectory and the corresponding control to cover the region over time, and possible changes in the environment are due to limited ranges of on-board sensors and to appearance of and/or motion of obstacles. Specifically, the problem is formulated as follows.

The authors are with the Department of Electrical and Computer Engineering, University of Central Florida, P.O. Box 162450, Orlando, FL 32816-2450, USA. Emails: yguo@ee.ucf.edu for Yi Guo, and qu@mail.ucf.edu for Zhihua Qu.

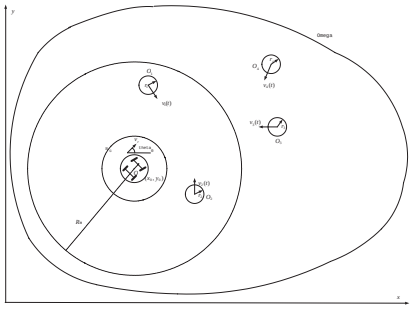

Fig. 1. Patrolling a connected region in the presence of dynamically moving obstacles

Assumption 1: *The robot under consideration is represented by a 2-dimensional circle with center at* $O(t) = (x, y)$ *and of radius* r_0*. Its motion is controlled but nonholonomic and is represented by the velocity vector* $v_r(t)$*.*

Assumption 2: *Set* Ω *to be covered is two-dimensional, connected (with respect to a circle of radius* r_0*), and has a convex shape.* T *is the time period to achieve the coverage.*

Assumption 3: *The range of robot's motion sensors is described by a circle centered at* $O(t)$ *and of radius* R_m*, while its coverage range (by its end-effector or detection sensors) is described by a circle centered at* $O(t)$ *and of radius* R_c*.*

Assumption 4: *The* i*th object,* $i = 1, \cdots, n_o$*, will be represented by a circle centered at point* $O_i(t)$ *and of radius* r_i*, denoted by* $B_i(O_i(t), r_i)$*. For moving objects, the origin* $O_i(t)$ *is time varying and moving with linear velocity vector* $v_i(t)$*, but velocity* v_i *is assumed to be a constant, denoted by* v_i^k*, within a specified period of time* $t \in [t_0 + kT_s, t_0 + (k+1)T_s)$ *(where* T_s *is often small).*

Patrolling Control Problem: *Given initial position* O_o *and initial orientation* θ_0 *of the robot and under assumptions 1 to 4, find a piecewise continuous steering control under which the robot moves collision-free and covers all the points in set* Ω *over time.* Mathematically, the problem is to determine a path $p(t)$ by ensuring

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

subject to

$$\|p(t) - O_i(t)\| \geq r_0 + r_i,$$
$$\forall t \in [t_0, t_0 + T] \text{ and } \forall i \in \{1, \cdots, n_o\}, \quad (2)$$

where $\phi(q,t)$ is a weighting function to be chosen by the designer.

Intuitively, the patrolling problem has at least one solution if the robot is capable of moving sufficiently fast, if motion of the obstacles does not make any part of Ω unconnected, and if those obstacles with $r_i > R_c$ are not stationary. The default value for $\phi(q,t)$ is 1. If $\phi(q^* \pm v, t) = \phi(q^*) = 2$ for any v satisfying $\|v\| = 0.5R_c$, path $p(t)$ either encircles q^* or covers q^* twice. If $\phi(q^*, t) = \phi(q^*) = \infty$, path $p(t)$ will pass through q^*. If $\phi(q,t) = \infty$ for all $t \neq t^*$, the path covers q at time $t = t^*$. For simplicity, the default value of $\phi(q,t) = 1$ is used in the following sections.

III. Patrolling Control Design

To tackle the proposed patrolling control problem, we first study the static complete coverage problem. And dynamic feasible trajectory is then designed with steering control explicitly constructed.

A. Complete Region Coverage

Since the coverage range of the robot is represented by a circle, the static coverage problem is to find a number of circles of radius R_c to completely cover Ω. Obviously, there are many solutions to this problem. Our objective is to find an optimal solution to minimize the repeated coverage. Therefore, the problem in concern is defined as:

Find a minimum number of circles of radius R_c to completely cover Ω in the two-dimensional plane.

1) Minimum Number of Circles to Cover A Rectangle: Minimal number of circles to cover a rectangle was reported in [6]. However, placement pattern of the circles to achieve the minimum number has not been seen in literatures. We present a solution to the optimal placement in the following.

To describe the solution, we refer the circle of radius R_c as a R_c-disk, and the rectangle to be covered is denoted by W. A pattern of R_c-strip is composed of a string of R_c-disks placed along a vertical line such that the distance between the centers of any two adjacent R_c-disks is $\sqrt{3}R_c$. We place m columns of R_c-strips oriented parallel to the y-axis with the distance between the centers of any two adjacent R_c-strips is $1.5R_c$. In a global cartesian coordinate with the origin is at the left bottom of the rectangle W, we place m R_c-strips parallel to the y-axis which contain n R_c-disks in each strip to completely cover the rectangular W, as shown in Figure 2. The center of the kth row ($1 \leq k \leq n$) and lth column ($1 \leq l \leq m$) disk is at $[x_c^{kl}, y_c^{kl}]$:

$$[x_c^{kl}, y_c^{kl}] = \begin{cases} \left[0.5 + (l-1)\frac{3}{2}R_c, (k-1)\sqrt{3}R_c\right], \\ \text{if } l \text{ is an odd integer;} \\ \\ \left[0.5 + (l-1)\frac{3}{2}R_c, \frac{\sqrt{3}}{2}R_c + (k-1)\sqrt{3}R_c\right], \\ \text{if } l \text{ is an even integer.} \end{cases} \quad (3)$$

The number of disks needed in each column and row, n and m, can be found as:

$$n = \begin{cases} Int\left(\frac{y_W}{\sqrt{3}R_c}\right) + 1, & \text{if } Rem\left(\frac{y_W}{\sqrt{3}R_c}\right) \leq \frac{1}{2}, \\ Int\left(\frac{y_W}{\sqrt{3}R_c}\right) + 2, & \text{if } Rem\left(\frac{y_W}{\sqrt{3}R_c}\right) > \frac{1}{2}; \end{cases} \quad (4)$$

$$m = \begin{cases} Int\left(\frac{x_W}{1.5R_c}\right) + 1, & \text{if } Rem\left(\frac{x_W}{1.5R_c}\right) \leq \frac{2}{3}, \\ \\ Int\left(\frac{x_W}{1.5R_c}\right) + 2, & \text{if } Rem\left(\frac{x_W}{1.5*R_c}\right) > \frac{2}{3}. \end{cases} \quad (5)$$

where Int is the integer operation and $Int(x)$ equals the integer part of x, $Rem(x) = x - Int(x)$, x_W is the length of the rectangular edge along the x-axis, y_W is the length of the rectangular edge along the y-axis.

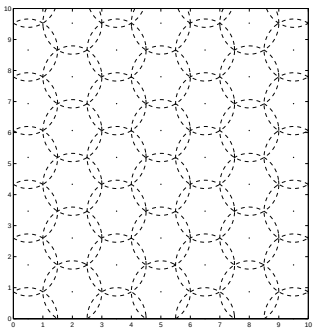

Fig. 2. Covering a rectangle using a minimum number of circles

Theorem 1: The disk placement pattern described above with centers of disks placed at (3) has a minimum number of disks to cover the rectangle W.

Proof: From (5), it can be readily obtained that the number of disks needed is:

$$N = \left(\frac{1}{\sqrt{3}}1.5\right)\left(\frac{y_W x_W}{R_c^2}\right) = \frac{2\sqrt{3}}{9}\left(\frac{y_W x_W}{R_c^2}\right) \quad (6)$$

The area covered by these disks is

$$A = \pi R_c^2 N = \frac{2\pi\sqrt{3}}{9}(y_W x_W) \quad (7)$$

Therefore, the ratio of the area to that of the rectangle (which may be thought of as measuring the proportion of unavoidable overlapping) is

$$d = \frac{A}{y_W x_W} = \frac{2\pi\sqrt{3}}{9} = 1.209 \quad (8)$$

It has been proved in [6] that 1.209 is the optimal value.

2) Minimum-Area Encasing Rectangle for an Arbitrary Closed Curve: In [7], a stepwise construction algorithm suitable for computer processing is presented to find the rectangle of minimum area in which a given arbitrary plane curve can be contained. It first utilizes the chain-coding scheme to represent an arbitrary plane curve in terms of a connected sequence of short, straight-line segments by overlaying a fine-spaced square grid and then connecting in sequence the grid notes that lie closest to the intersections of the curve with the grid. Then in the chain code, the octal digits 0 to 7 are used to represent different directions in the counter-clockwise sense beginning with the positive x direction. The minimum-perimeter convex polygon that encases the convex hull of the given curve is then determined. And finally the minimum-area rectangle that encases this convex polygon is found. It was proved that the obtained rectangle is the minimum-area encasing one for the given curve.

3) Minimum Number of Disks for Region Coverage: Based on the above subsections, we present an algorithm for placing a minimal number of disks to completely cover the region Ω.

Step 1: Determine a minimum-area rectangle that encases the boundary of the Ω using the algorithm presented in Section III-A.2.

Step 2: Rotate the global cartesian coordinate so that the origin is at the lower-left corner of the rectangle and two edges of the rectangle is on the positive x and y axes respectively.

Step 3: Place disks using the pattern described in Section III-A.1 to cover the rectangle generated in the previous step.

Step 4: Exclude the disks whose centers are outside the region, and denote the centers of all rest disks as the set of points A:

$$A: \qquad (x_i, y_i), i = 1, \ldots, n_A \tag{9}$$

where n_A is the total number of points in the set A, and (x_i, y_i) are placed according to the pattern of (3).

Note that in Step 4, some points near the boundary may not be covered by the disks whose centers are inside Ω. An additional step may be added for a better coverage for points near the boundary: Check the distance between each vertex and the set A; if the smallest distance is bigger than the radius of the disk R_c, then add the vertex to the set A. Additional analysis to guarantee near-boundary-point coverage is out of the scope of this paper.

Figure 3 shows a convex region being covered by minimum number of disks.

B. Patrolling Paths

After the coverage disks are placed as shown in Figure 3, a patrolling path can be planned for the robot to go through the centers of the disks, *i.e.*, the set A. It was presented in [4] that complete coverage paths can be searched based on cell representation of the environment according to different rules. Intuitively, a path exists to go along the boundary

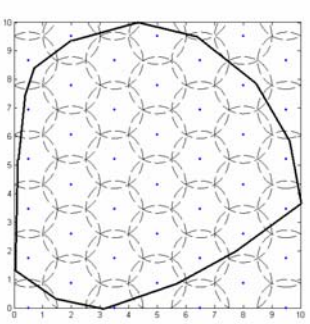

Fig. 3. Covering a convex region using a minimum number of disks

of the region in a spital. We first define the following terminologies:

Definition 1: For each point $(x_i, y_i) = (x_c^{kl}, y_c^{kl})$ in the set A, its six neighboring points are at
$(x_c^{(k+1)l}, y_c^{(k+1)l}), (x_c^{(k-1)l}, y_c^{(k-1)l})$,
$(x_c^{k(l+1)}, y_c^{k(l+1)}), (x_c^{k(l-1)}, y_c^{k(l-1)})$,
$(x_c^{(k-1)(l+1)}, y_c^{(k-1)(l+1)}), (x_c^{(k-1)(l-1)}, y_c^{(k-1)(l-1)})$.
In other words, its neighbors are the vertices of the hexagon whose center is at the point.
Figure 4 shows the six neighbors of a point.

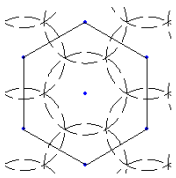

Fig. 4. Six neighbors of a point

Definition 2: Assume that the boundary is represented by a set of points B. The distance of a point (x, y) to the boundary, denoted by $dist((x, y), B)$, is the smallest distance of (x, y) to all points of B. That is,

$$dist((x, y), B) \quad = \quad \min_{(a,b)\in B} \|(x, y), (a, b)\| \tag{10}$$

where $\|(x, y), (a, b)\|$ denotes the Euclidean distance of the two points (x, y) and (a, b).

Our algorithm to find a complete coverage path is as follows:
Set Start point to Current
Set all other points in A to Unvisited
LOOP
Find Unvisited Neighboring point whose distance to the boundary is the smallest
If no Neighbor point found then Mark as Visited and Stop at End
Mark as Visited and set Current point to Neighboring point
LOOP END

Figure 3 shows such a path covering a convex region. The path is represented by a sequence of points, denoted by:

$$P: \qquad (x_j, y_j), j = 1, \ldots, n_A. \tag{11}$$

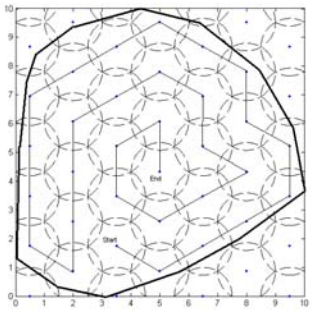

Fig. 5. A coverage path

Note that the difference of the set P and A is that the points in P are in sequence while the points in A are not. In other words, (x_1, y_1) in P is the Start position of the robot, and (x_{n_A}, y_{n_A}) in P is the End position.

C. Feasible Patrolling Trajectories for Nonholonomic Robots

It is well known that nonholonomic constraints of mobile robots (kinematic constraints) make time derivatives of configuration variables of the system non-integrable, and any given path in the configuration space does not necessarily correspond to a feasible path for the nonholonomic system. To make the planned path trackable by nonholonomic robots, we have to design feasible trajectories which accounts for the kinematic constrains of the robots, and are also free of collision to obstacles in the environment. We apply the recently developed real time trajectory generation algorithm, published in [5], to connect every two adjacent path points in the set P. To make the paper self-complete, we cite the main result from [5].

For a car-like mobile robot whose front wheels are steering wheels and rear wheels are driving wheels but have a fixed forward orientation, the state space representation of the kinematic model taking nonholonomic constrains is given by:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} 1 & 0 & -\frac{l}{2}\sin(\theta) & 0 \\ 0 & 1 & \frac{l}{2}\cos(\theta) & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \rho\cos(\theta) & 0 \\ \rho\sin(\theta) & 0 \\ \frac{\rho}{l}\tan\phi & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \quad (12)$$

where $q = [x\ y\ \theta\ \phi]$ is the state vector: $[x\ y]$ represents the Cartesian coordinates of the guide point, θ is the orientation of the robot body with respect to the x-axis, ϕ is the steering angle; l is the distance between the two wheel-axle centers, ρ is the radius of back driving wheel; u_1 is the angular velocity of the driving wheel, and u_2 is the steering rate of front guiding wheel. The range of ϕ is limited to be within $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ due to singularity, and θ is within $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ to ensure an one-to-one map of the coordinate transformation.

The kinematic model (12) can be transformed into the well-defined chained-form:

$$\begin{aligned} \dot{z}_1 &= v_1 \\ \dot{z}_2 &= v_2 \\ \dot{z}_3 &= z_2 v_1 \\ \dot{z}_4 &= z_3 v_1 \end{aligned} \quad (13)$$

under the following coordinate and input transformations:

$$\begin{aligned} z_1 &= x - \frac{l}{2}\cos(\theta) \\ z_2 &= \frac{\tan(\phi)}{l\cos^3(\theta)} \\ z_3 &= \tan(\theta) \\ z_4 &= y - \frac{l}{2}\sin(\theta), \end{aligned} \quad (14)$$

$$\begin{aligned} u_1 &= \frac{v_1}{\rho\cos(\theta)} \\ u_2 &= -\frac{3\sin(\theta)}{l\cos^2(\theta)}\sin^2(\phi)v_1 \\ &\quad + l\cos^3(\theta)\cos^2(\phi)v_2. \end{aligned} \quad (15)$$

Based on the main results of [5], we give our patrolling control algorithm as follows:

Considering every adjacent pair of points in P, take the robot configuration at the points as boundary conditions $q^0 = [x_0, y_0, \theta_0, \phi_0]^T$ and $q^f = [x_f, y_f, \theta_f, \phi_f]^T$ with $\phi_0 = \phi_f = 0$. If satisfying the condition that $x_0 - \frac{l}{2}\sin\theta_0 \neq x_f - \frac{l}{2}\sin\theta_f$, and that $|\theta_0 - \theta_f| < \pi$, a collision-free path can be generated analytically by undertaking following steps:

(i) Select coordinates (x, y) of the working space such that $\theta \neq \frac{\pi}{2}$, apply state and input transformations (14) and (15), determine the corresponding boundary conditions $z^0 = [z_1^0, z_2^0, z_3^0, z_4^0], z^f = [z_1^f, z_2^f, z_3^f, z_4^f]$, and obtain the dynamics in the chained form (13).

(ii) 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, that the centers of objects O_i are located at (x_i^k, y_i^k) at $t = t_0 + kT_s$, and that these objects are all moving with known constant velocities $v_i^k \triangleq [\ v_{i,x}^k\ v_{i,y}^k\ ]^T$ for $t \in [t_0^j + kT_s^j, t_0^j + (k+1)T_s^j)]$. Then, for $k = 0, \cdots, \overline{k} - 1$, determine recursively constants a_6^k by ensuring the following second-order inequality (or inequalities): $\forall i \in \{1, \cdots, n_o\}$

$$\begin{aligned} \min_{t\in[\underline{t}_i^*, \overline{t}_i^*]} & g_2(z_1(t), k)(a_6^k)^2 + g_{1,i}(z_1(t), k, \tau)a_6^k \\ & + g_{0,i}(z_1(t), k, \tau)|_{\tau = t - t_0^j - kT_s^j} \geq 0, \end{aligned} \quad (16)$$

where $[\underline{t}_i^*, \overline{t}_i^*] \subset [t_0^j + kT_s^j,\ T_j]$ is the time interval (if exists*) during which

$$x_i^k \in [z_1(t) - v_{i,x}^k\tau - r_i - R,\ z_1(t) - v_{i,x}^k\tau + 0.5l + r_i + R].$$

*If the interval does not exist for some or all i, inequality (16) is not needed for those objects.

In (16), functions $z_1(t)$, $g_2(\cdot)$, $g_{1,i}(\cdot)$ and $g_{0,i}(\cdot)$ are defined as follows:

$$z_1(t) = z_1^k + \frac{z_1^f - z_1^0}{T}(t - t_0^j - kT_s^j) \quad \forall t \in [t_0^j + kT_s^j,\ T_j];$$

$$x_i^0 = x_i(t_0^j), \quad y_i^0 = y_i(t_0^j); \quad x_i^k = x_i^0 + T_s^j \sum_{j=0}^{k-1} v_{i,x}^j,$$

$$y_i^k = y_i^0 + T_s^j \sum_{j=0}^{k-1} v_{i,y}^j, \quad \text{if } k > 0;$$

$$z_4^0 = y_0 - \frac{l}{2}\sin(\theta_0), \quad z_3^0 = \tan(\theta_0), \quad z_2^0 = 0,$$

$$z_2^k = z_2^{k-1} + \int_{t_0^j + (k-1)T_s^j}^{t_0^j + kT_s^j} v_2(t)dt,$$

$$z_3^k = z_3^{k-1} + \frac{z_1^f - z_1^0}{\overline{k}} \int_{t_0^j + (k-1)T_s^j}^{t_0^j + kT_s^j} \int_{t_0^j + (k-1)T_s^j}^{s} v_2(t)dtds,$$

$$z_1^k = z_1^0 + \frac{k(z_1^f - z_1^0)}{\overline{k}}, \quad \text{if } k > 0; \qquad (17)$$

$$Y^k = \begin{bmatrix} z_4^k \\ z_3^k \\ z_2^k \\ y_f - \frac{l}{2}\sin(\theta_f) \\ \tan(\theta_f) \\ 0 \end{bmatrix}, \quad A^k = \begin{bmatrix} (z_1^k)^6 \\ 6(z_1^k)^5 \\ 30(z_1^k)^4 \\ (z_1^f)^6 \\ 6(z_1^f)^5 \\ 30(z_1^f)^4 \end{bmatrix},$$

$$B^k = \begin{bmatrix} 1 & z_1^k & (z_1^k)^2 & (z_1^k)^3 & (z_1^k)^4 & (z_1^k)^5 \\ 0 & 1 & 2z_1^k & 3(z_1^k)^2 & 4(z_1^k)^3 & 5(z_1^k)^4 \\ 0 & 0 & 2 & 6z_1^k & 12(z_1^k)^2 & 20(z_1^k)^3 \\ 1 & z_1^f & (z_1^f)^2 & (z_1^f)^3 & (z_1^f)^4 & (z_1^f)^5 \\ 0 & 1 & 2z_1^f & 3(z_1^f)^2 & 4(z_1^f)^3 & 5(z_1^f)^4 \\ 0 & 0 & 2 & 6z_1^f & 12(z_1^f)^2 & 20(z_1^f)^3 \end{bmatrix}$$

and

$$\underline{f}(z_1(t)) = \left[1\ z_1(t)\ (z_1(t))^2\ (z_1(t))^3\ (z_1(t))^4\ (z_1(t))^5\right],$$
$$g_2(z_1(t), k) = \left[(z_1(t))^6 - \underline{f}(z_1(t))(B^k)^{-1}A^k\right]^2,$$
$$g_{1,i}(z_1(t), k, \tau) = 2\left[(z_1(t))^6 - \underline{f}(z_1(t))(B^k)^{-1}A^k\right] \cdot \left[\underline{f}(z_1(t))(B^k)^{-1}Y^k - y_i^k - v_{i,y}^k\tau\right],$$
$$g_{0,i}(z_1(t), k, \tau) = \left[\underline{f}(z_1(t))(B^k)^{-1}Y^k - y_i^k - v_{i,y}^k\tau\right]^2 + (z_1(t) - x_i^k - v_{i,x}^k\tau)^2 - (r_i + R + 0.5l)^2.$$

(iii) A feasible, collision-free path in the transformed state has the form

$$z_4(z_1) \quad = \quad F(z_1) = a^k f(z_1) \qquad (18)$$

where a^k is solved according to

$$a^k = \left[\ a_0^k \quad a_1^k \quad a_2^k \quad a_3^k \quad a_4^k \quad a_5^k \quad a_6^k\ \right],$$
$$\left[a_0^k, a_1^k, a_2^k, a_3^k, a_4^k, a_5^k\right]^T = (B^k)^{-1}(Y^k - A^k a_6^k). \qquad (19)$$

(iv) The steering inputs to achieve path (18) are given by, for $t \in (t_0^j + kT_s^j, t_0^j + (k+1)T_s^j]$,

$$\begin{aligned} v_1(t) &= v_1 = \frac{z_1^f - z_1^0}{T}, \\ v_2(t) &= 6[a_3^k + 4a_4^k z_1^k + 10a_5^k (z_1^k)^2 + 20a_6^k (z_1^k)^3]v_1 \\ &+ 24[a_4^k + 5a_5^k z_1^k + 15a_6^k (z_1^k)^2](t - t_0^j - kT_s^j)v_1^2 \\ &+ 60(a_5^k + 6a_6^k z_1^k)(t - t_0^j - kT_s^j)^2 v_1^3 \\ &+ 120a_6^k (t - t_0^j - kT_s^j)^3 v_1^4. \end{aligned} \qquad (20)$$

(v) The corresponding feasible, collision-free Cartesian trajectory is given by

$$y \quad = \quad F(x - 0.5l\cos(\theta)) + 0.5l\sin(\theta), \qquad (21)$$

where θ can be found in closed form from state transformation (14) under steering inputs (20) and control mapping (15).

IV. Conclusions

We have studied the patrolling control problem, which is defined to find a piecewise continuous steering control under which the robot moves collision-free and covers all the points in a convex set over time. Constructive algorithms are given in sequential modules to solve the problem. First, we place a minimum-area rectangle that encases the boundary of the set. Second, minimum number of circles of the radius of coverage range are placed to completely cover the rectangle. Third, a patrolling path is searched along the boundary of the set in a spital. Feasible trajectories are then designed to account for the nonholonomic kinematics of the robot and the dynamic obstacles detected by the robot onboard sensors. Since analytic solutions are given in generating feasible trajectories, the algorithm can be implemented in real time. Future work exists to refine the design in the last step and to enable performance evaluation.

References

[1] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2002, pp. 1327–1332.

[2] R. N. de Carvalho, H. A. Vidal, P. Vieira, and M. I. Ribeiro, "Complete coverage path planning and guidance for cleaning robots," in *Proceedings of IEEE Int. Symposium on Industrial Electronics*, 1997.

[3] Y. Guo, L. E. Parker, and R. Madhavan, "Towards collaborative robots for infrastructure security applications," in *Proceedings of The 2004 International Symposium on Collaborative Technologies and Systems*, San Diego, CA, Jan. 2004, pp. 235–240.

[4] A. Zelinsky, R. A. Jarvis, J. C. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environments by a mobile robot," in *Proceedings of IEEE International Conference on Robotics and Automation*, 1993, pp. 2612–2619.

[5] Z. Qu, J. Wang, and C. E. Plaisted, "A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles," in *Proceedings of 2003 Florida Conference on Recent Advances in Robotics*, May 2003, also to appear in IEEE Transactions on Robotics.

[6] R. Kershner, "The number of circles covering a set," *Amer. J. Math.*, vol. 61, pp. 665–671, 1939.

[7] H. Freeman and R. Shapira, "Determining the minimum-area encasing rectangle for an arbitrary closed curve," *Communications of the ACM*, vol. 18, no. 7, pp. 409–413, 1975.