Dynamic Perimeter Surveillance with a Team of Robots

David Saldaña[1], Reza Javanmard Alitappeh[2], Luciano C. A. Pimenta[2], Renato Assunção[1], and Mario F. M. Campos[1]

***Abstract*— In this paper, we propose a motion planning method to escort a set of agents from one place to a goal in an environment with obstacles. The agents are distributed in a finite area, with a time-varying perimeter, in which we put multiple robots to patrol around it with a desired velocity. Our proposal is composed of two parts. The first one generates a plan to move and deform the perimeter smoothly, and as a result, we obtain a twice differentiable boundary function. The second part uses the boundary function to compute a trajectory for each robot, we obtain each resultant trajectory by first solving a differential equation. After receiving the boundary function, the robots do not need to communicate among themselves until they finish their trajectories. We validate our proposal with simulations and experiments with actual robots.**

I. Introduction

In the research area of perimeter surveillance, the problem of using multiple robots to patrol around a static region has received much attention in the literature [1], [2], [3], [4]. In some cases, the guarded region contains multiple agents, like humans or animals, that must be protected. When those agents need to be escorted by a team of robots from one place to a goal, the classical perimeter surveillance techniques are not prepared for such a task, since they do not take into account the dynamics of the perimeter. We call this concept *dynamic perimeter surveillance*, which brings remarkable new challenges, like the necessity of deformations in the perimeter or shape adaptation based on the configuration of the environment. While robots keep moving through the perimeter in counterclockwise manner, they create a virtual fence to surround and protect the internal agent(s). When multiple robots circulate around a specific area with high speed, they reduce the possibility to allow external agents to get inside. By the same way, this behavior also prevents the internal agents to get outside. Dynamic perimeter surveillance covers applications like: the protection of a group of people that must be escorted from one place to a goal; shepherding of animals; large area monitoring and mapping; or multi-robot pattern generation.

In [5], the authors presented a decoupled controller to move a robotic swarm collectively as a group. Simple shapes like square and ellipse are adapted to move the group through obstacles. A stable method that can maintain a minimum distance among the robots, while they form a desired dynamic region, it is presented in [6]. The same authors in [7] modeled elaborated shapes by joining simple functions. These works offer adaptable behaviors for the shapes, but they do not focus on leaving the internal region available.

In this paper we focus on planning the motion of a team of robots to guard a region by circulating along its boundary and leaving the internal area available. In other works that also deal with the problem of moving single robot [1] or multiple robots [3] to track a curve, usually the curve is static or changes according to a given function of time without any input from the environment where the robots are moving. Some other works like [8] consider also the case were it is necessary to estimate a dynamic boundary defined in the environment. In contrast, in this work we are interested in the integration between robot motion planning and boundary motion planning. It is also our objective to plan the motion of the perimeter.

If the movement of our deformable shape is considered to be analogous to the movement of a robot, then the planning problem can be approached by adapting classical motion planning techniques in robotics. Sample-based planner techniques have been frequently used to compute collision free movements in unstructured environments, specially for systems with high degrees of freedom [9]. Rapidly Random Tree (RRT) [10] has received a great deal of attention due to its ability to perform dense exploration of the whole configuration space in a more practical and faster way [11], [12], [13]. RRT is also applied for kinodynamic planning in which control inputs are planned to drive a robot from an initial to a final state [14]. This technique works well in high dimensional state spaces, for instance, in [15], they applied a RRT motion planner to control an acceleration limited manipulator. In our work, we propose a simplified RRT-Kinodynamic technique, where the vertices in the tree are expanding progressively by applying a random control input. Thus we do not need a local planner in contrast to the related techniques.

Our contributions are threefold. First, we propose a model that includes the time-variant factor in *perimeter surveillance*. Second, we adapted a classical motion planning technique to obtain an analytical function that generates a smooth changing behavior for a given shape. This behavior can result in simultaneous transformations like translation, shrinking and expansion in order to guard and to move a desired region in a non-discretized environment. Third, we take advantage of the calculated boundary function to develop a method that maintains multiple robots circulating around a dynamic perimeter with a desired constant velocity.

II. Problem Statement

Consider a compact environment $\mathcal{E} \subset \mathbb{R}^2$, which contains a set of obstacles $\mathcal{O} \subset \mathcal{E}$. The goal is to use a set of robots $\mathcal{R}$ to patrol along the boundary of a deformable region $\Omega \subset \mathcal{E}$

[1]D. Saldaña, R. Assunção, and M.F.M. Campos are with the Computer Vision and Robotics Laboratory (VeRLab), Computer Science Department, UFMG. E-mails: {saldana, assuncao, mario}@dcc.ufmg.br

[2] R. Javanmard and L. C. A. Pimenta are with the Graduate Program in Electrical Engineering (PPGEE), UFMG. E-mails: {Rezajavanmard, lucpim}@cpdee.ufmg.br

Universidade Federal de Minas Gerais (UFMG), MG, Brazil.

*The authors gratefully acknowledge the support of CAPES, CNPq and FAPEMIG. D. Saldaña also thanks to COLCIENCIAS for its support.

while this region moves from an initial point to a final point without colliding with any obstacle. The boundary/perimeter of Ω is denoted by $\partial\Omega$ that can be modeled by a parametric shape:

Definition 1 (parametric shape): The **parametric shape** $\Phi : [0,1] \times \mathcal{Q} \rightarrow \mathcal{E}$, where $\mathcal{Q}$ is the n-dimensional configuration space of the shape. So that, Φ is a function that maps the connected set $\partial\Omega$ into $\mathcal{E}$ and can transform its shape based on n parameters as degrees of freedom.
The free parameters of Φ can change properties such as position, orientation, and size. Given these n degrees of freedom, the configuration space of the fixed boundary is a compact space and the configuration vector is given by $\mathbf{q} = [q_1, q_2, ..., q_n]^T, \mathbf{q} \in \mathcal{Q}$.

For instance, a shape with the form of an ellipse with fixed orientation has the following equation,

$$\Phi(s, \mathbf{q}) = \begin{bmatrix} q_0 \cos(2\pi s) + q_2 \\ q_1 \sin(2\pi s) + q_3 \end{bmatrix}, \tag{1}$$

for all $s \in [0,1]$, where the parameters q_0 and q_1 are the semi-axes of the ellipse and the parameters q_2 and q_3 are the translations in coordinates x and y respectively.

Additionally, we represent the free configuration space as $\mathcal{Q}_{free} = \mathcal{Q} \backslash \mathcal{QO}$, where $\mathcal{QO}$ is the obstacle set $\mathcal{O}$ in the configuration space. We define a collision free trajectory as a function $\tau : [t_0, t_f] \rightarrow \mathcal{Q}_{free}$ that maps a finite time interval $[t_0, t_f]$ into $\mathcal{Q}_{free}$ starting at the initial configuration $\mathbf{q}_0$ and ending at the final configuration $\mathbf{q}_f$. The shape dynamics along the trajectory is defined as follows:

Definition 2 (boundary function): The **boundary function** $\gamma : [0,1] \times \mathbb{R} \rightarrow \mathcal{E}$ describes how the shape Φ changes its configuration $\mathbf{q}$ along time $t \in [t_0, t_f]$, and satisfies the equation,

$$\gamma(t, s) = \Phi(s, \tau(t)), \tag{2}$$

which is twice differentiable.

We have divided our objective into two main problems to be solved. The fist one is about planning the movement of the whole perimeter, stated as follows:

*Problem 1 (**Motion planning for the boundary**):* Given a deformable shape Φ, how to compute a trajectory τ to obtain the dynamic boundary γ, which can change its shape smoothly in order to move from an initial configuration $\mathbf{q}_0$ to a final configuration $\mathbf{q}_f$ without leaving $\mathcal{Q}_{free}$.

Once the dynamics of the boundary is identified, the second problem aims to identify the trajectory for each robot:

*Problem 2 (**Motion planning for robots**):* Given a boundary function γ, how to calculate the trajectory of a team of robots, so that each robot circulates the time-varying boundary with a desired velocity v in a counterclockwise manner.

We approach Problems 1 and 2 in Sections III and IV, respectively.

III. Motion Planning for the Boundary

In order to solve Problem 1, we propose a method to find a smooth analytical function τ that defines the transformation of Φ to reach to the goal $\mathbf{q}_f$. We assume that the boundary parameters can evolve according to a double integrator with dynamics given by: $\ddot{\mathbf{q}} = \mathbf{u}$, where $\mathbf{u} \in \mathcal{U}$ is a control input in the compact space $\mathcal{U} \subset \mathbb{R}^n$. The state space for the boundary is defined by $\mathcal{X} = \mathbb{R}^{2n}$ and the state vector is denoted by $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^T \in \mathcal{X}$. In order to generate the trajectory τ from an initial state $\mathbf{x}_0 = [\mathbf{q}_0, \dot{\mathbf{q}}_0]^T$ to a goal state $\mathbf{x}_f = [\mathbf{q}_f, \dot{\mathbf{q}}_f]^T$, the motion equation can be defined as:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{3}$$

where f is a continuous function, $\mathbf{x}(t) \in \mathcal{X}$ in time $t \in \mathbb{R}_{>0}$.

We create the motion plan by extending the standard kinodynamic motion planning technique [14], which is based on the RRT (Rapidly Random Tree) [10]. In Algorithm 1, we present our version of the kinodynamic motion planner. Let $G = \{V, E\}$ be a tree which is composed of a set V of n vertices and a set E of $n-1$ edges. Each vertex $v \in V$contains its current state $\mathbf{x}_v$ and a quadratic function τ_v that describes how to arrive from the ancestor of v to v within a finite time interval.

Algorithm 1: *Kinodynamic boundary motion planner*

```
   Input: (E, O), (x_0, x_f) // Input map, desired states.
   Output: τ // Motion function for the shape.
1  x_r ← x_0 // Set initial state to x_r.
2  G ← Extend(G, (x_0, 0)) // Add initial state to the tree.
   // Until a termination criteria is satisfied.
3  while (! Termination_Criteria) do
4      u ← RandomSample(U) // random control input.
       // Compute new state by integrating u.
5      x', τ_t = Compute_NewState(u, x_r)
6      if Is_Collision_Free(E, O, x') then
           // Add new state to the tree.
7          G ← Extend(G, (x', τ_t))
           // If the goal is achieved.
8          if Is_Close(x', x_f) then
               // Create a trajectory function.
9              τ ← Construct_Trajectory(G, x_0, x_f)
10             Return τ
           end
       end
       // Choose a vertex randomly or using a heuristic
11     x_r ← Select_Vertex(G)
   end
```

In this algorithm, we initially create a graph with a single vertex that represents the initial state (lines 1,2). In line 4, we generate a random acceleration vector $\mathbf{u} = \ddot{\mathbf{q}}$, which in line 5, we integrate in order to obtain the new state vector $\mathbf{x}' = [\mathbf{q}', \dot{\mathbf{q}}']^T$. We validate the feasibility of the new state $\mathbf{x}'$ in line 6, then the tree G is extended to add a vertex with the new information $(\mathbf{x}', \tau_t)$ in line 7. We check if the new state is close enough to the goal state in line 8. This process (lines 3-10) will be repeated by choosing vertices in the tree (line 11), until the goal state is approached (line 8) or a termination criterion is met (line 3). Finally, if the new state $\mathbf{x}'$ is close to the desired state $\mathbf{x}_f$, we obtain a vertex with $m-1$ ancestors.

The output of this iterative process is a twice-differentiable function $\tau(t)$ that is calculated in line 9, which has the

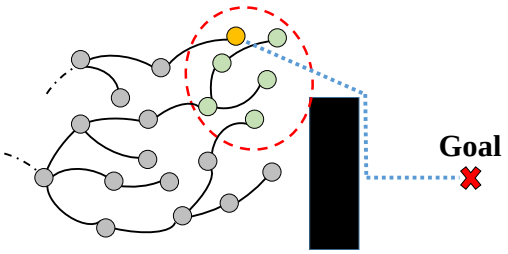


Fig. 1. A set of the closest vertices to the goal are highlighted inside the dashed ellipse. The dotted line indicates to the geodesic path from the selected vertex to the goal.

following form:

$$\tau(t) = \begin{cases} \tau_1(t) & \text{if } t_0 \leq t < t_1 \\ \tau_2(t) & \text{if } t_1 \leq t < t_2 \\ \vdots & \\ \tau_m(t) & \text{if } t_{m-1} \leq t < t_m, \end{cases}$$

where each function $\tau_i(t)$, $i = 1, 2..., m$ is the calculated quadratic function calculated in line 5,

$$\tau_i(t) = \begin{bmatrix} a_1 t^2/2 + b_1 t + c_1 \\ a_2 t^2/2 + b_2 t + c_2 \\ \vdots \\ a_n t^2/2 + b_n t + c_n \end{bmatrix}.$$

The coefficients a_j, b_j, c_j, $j = 1, 2..., n$ are computed by the integration of $\mathbf{u}$ in $t \in [t_{j-1}, t_j]$ (line 5), considering that $\mathbf{u}$ is constant during the time interval. As t_m is our final time, we have $t_f = t_m$. Therefore, we estimate the boundary function γ by inserting the calculated τ in Eq. (2). We can argue that the shape Φ has a continuous smooth motion because γ changes quadratically with respect to time $t \in [t_0, t_f]$. This was the reason for choosing the double integrator model for the boundary.

For the vertex selection of line 11, instead of a simple random vertex choice, we use a bias toward the goal to speed up the algorithm's convergence. Then, based on a random variable $\alpha \in [0, 1]$, we choose a random vertex with probability α or we use a heuristic method with probability $(1 - \alpha)$. The heuristic is based on the geodesic distance between the vertices and the goal. We choose a vertex randomly from the set of the $k \in \mathbb{N}$ nearest nodes to the goal, where "nearest" means the ones closest to the goal according to te geodesic distance. Fig. 1 illustrates a random tree, where a set of $k = 6$ candidate vertices is highlighted to select one of them. Additionally, in contrast to the standard RRT techniques, we do not need a local planner to connect a sample point to a node. This is done by setting a small expanding step and checking the feasibility of the new state. This heuristic has a remarkable advantage, it only requires to compute the map distance for $\mathbb{R}^2$.

IV. Motion Planning for the Robots

In this section, we address Problem 2 by computing the trajectory of each robot that circulates with a desired velocity around the boundary. Let $\mathcal{R} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_k\}$ be the set of configurations of the k robots, where each robot configuration is represented by the vector $\mathbf{r}_i = [x_i, y_i]^T$,

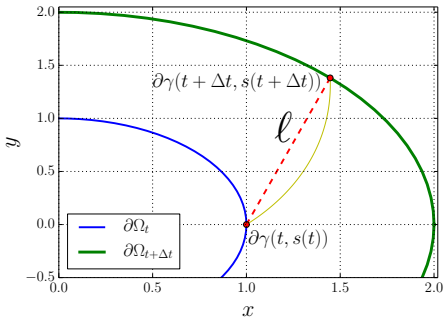


Fig. 2. Robot trajectory (yellow line) in a time interval $[t, t + \Delta t]$, it is approximated by a linear movement (dashed line) for a Δt. The blue curve is the boundary at time t, and the green curve is the boundary at time $t+\Delta t$. The robot moves from point $\gamma(t, s(t))$ to point $\gamma(t + \Delta t, s(t + \Delta t))$.

$i = 1, 2, ..., k$ where x_i, y_i define the coordinates in the Euclidean space. We assume the holonomic fully actuated model for each robot:

$$\dot{\mathbf{r}}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = \mathbf{w}_i, \tag{4}$$

where $\mathbf{w}_i$ is velocity inputs. Thus the velocity magnitude is given by $v_i = \sqrt{\dot{x}_i^2 + \dot{y}_i^2}$. In order to track the perimeter $\partial\Omega$, each robot $\mathbf{r}_i$ circulates along the dynamic boundary in counter-clockwise manner with a desired velocity v. At $t = t_0$, the robot $\mathbf{r}_i$ starts at location $\gamma(t_0, s_0)$, where the constant $s_0 \in s$ maps to an arbitrary point of the boundary. Now we define a function describing the trajectory of each robot along the boundary.

Definition 3: The **trajectory of robot** i along the boundary is represented by the function $S_i(t)$, where for each time t, there is a function $s(t) : \mathbb{R}_{\geq 0} \to [0, 1]$ such that

$$S_i(t) = \gamma(t, s(t)), \ \forall t > t_0$$

in which, $s(t)$ maps the curve parameter s to the robot position. It satisfies the initial condition $s_0 = s(t_0)$.

Theorem 1: If a robot is moving with a desired velocity v in counter-clockwise direction along a dynamic boundary, described by the function $\gamma(t, s)$, then the robot trajectory $S(t) = \gamma(t, s(t))$ that starts at a point in $\gamma(t_0, s_0)$, is such that $s(t)$ satisfies the ordinary differential equation

$$\dot{s} = \frac{1}{x_s^2 + y_s^2}\left(-x_s x_t - y_s y_t + \sqrt{v^2(x_s^2 + y_s^2) + B}\right), \tag{5}$$

where $B = -x_s^2 y_t^2 + 2x_s x_t y_s y_t - x_t^2 y_s^2$, and $x_s = \frac{\partial x}{\partial s}$, $y_s = \frac{\partial y}{\partial s}$, $x_t = \frac{\partial x}{\partial t}$, and $y_t = \frac{\partial y}{\partial t}$.

Proof: We analyze the movement between time t and time $t + \Delta t$. It may be approximated by a straight linear movement for a very small Δt. Then between t and $t + \Delta t$, the robot moves a distance ℓ, as illustrated in Figure 2.

In this proof, we abuse the notation of x and y in order to represent separate coordinates of γ. Then we can rewrite the γ function as a combination of functions x, and y,

$$\gamma(t, s) = \begin{bmatrix} x(t, s) \\ y(t, s) \end{bmatrix}.$$

The distance ℓ is calculated by the Pythagorean theorem,

$$\ell^2 = \Delta x^2 + \Delta y^2,$$

where $\Delta x = x(t+\Delta t, s(t+\Delta t)) - x(t, s(t))$ and $\Delta y = y(t+\Delta t, s(t+\Delta t)) - y(t, s(t))$. As the robot moves linearly with constant velocity v, then $\ell = v\Delta t$,

$$v^2\,\Delta t^2 = \Delta x^2 + \Delta y^2,$$

which is the same in Newton's notation,

$$v^2 = \dot{x}^2 + \dot{y}^2. \tag{6}$$

Using the chain rule for $\dot{x}$ and $\dot{y}$, we obtain $\dot{x} = x_t + x_s\dot{s}$ and $\dot{y} = y_t + y_s\dot{s}$, respectively. Replacing $\dot{x}$ and $\dot{y}$ in Equation (6),

$$v^2 = \Big(x_t + x_s\dot{s}\Big)^2 + \Big(y_t + y_s\dot{s}\Big)^2.$$

Expanding terms and simplifying we obtain,

$$v^2 = x_t^2 + 2\dot{s}x_sx_t + \dot{s}^2x_s^2 + y_t^2 + 2\dot{s}y_sy_t + \dot{s}^2y_s^2.$$

Solving for $\dot{s}$, we obtain two solutions. As the robot is moving in counter-clockwise manner, the variable s is always increasing, then the solution is

$$\dot{s} = \frac{1}{x_s^2 + y_s^2}\left(-x_sx_t - y_sy_t + \sqrt{v^2(x_s^2+y_s^2) + B}\right),$$

where $B = -x_s^2y_t^2 + 2x_sx_ty_sy_t - x_t^2y_s^2$. The robot's trajectory is computed by the function $S(t) = \gamma(t, s(t))$, that satisfies the initial condition $s_0 = s(t_0)$. ■

The first-degree differential equation above can be solved by numerical methods like Runge-Kutta4. Then, the starting point $\gamma(t_0, s_0)$ can be used as the initial condition in order to obtain an approximation of the function $s(t)$. Hence, each robot $\mathbf{r}_i$, $i = 1, 2, ..., k$ receives the boundary function γ with a different initial location $s_i \in [0, 1]$ in order to obtain its trajectory $S_i(t)$, by solving the Differential Equation (5) in a distributed way. After receiving the boundary function, no robot needs to communicate with other robots until it has completed its trajectory.

In order to enforce each robot $\mathbf{r}_i$ to follow the desired trajectory $S_i(t)$, we compute the control inputs in (4) by means of:

$$\mathbf{w}_i = \dot{S}_i(t) + k(S_i(t) - \mathbf{r}_i(t)), \tag{7}$$

where $k > 0$ is a constant.

Minimum linear velocity to track the boundary

The minimum linear velocity, v_{min}, required to make the robot advance along the curve must satisfy $\dot{s} > 0\ \forall\ t \in [t_0, t_f]$. Using Equation (5) with $\dot{s} > 0$,

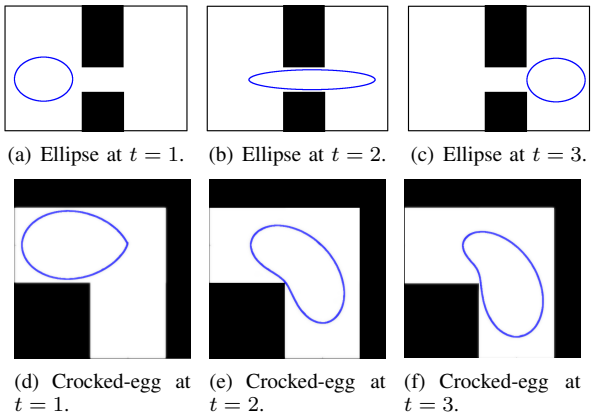
$$\frac{1}{x_s^2 + y_s^2}\left(-x_sx_t - y_sy_t + \sqrt{v^2(x_s^2+y_s^2) + B}\right) > 0,$$

and isolating v, we obtain $v > \sqrt{x_t^2/2 + y_t^2/2}$. As the right side of the Equation depends on time t, and the velocity is constant, the minimum constant velocity that can walk around the boundary is the maximum in the interval:

$$v_{\min} > \max\left(\sqrt{\frac{x_t^2}{2} + \frac{y_t^2}{2}}\right). \tag{8}$$

For example, for the growing circle with boundary equation $\gamma(t, s) = [t\cos(s), t\sin(s)]^T$, by applying Equation (8),

$$v_{\min} > \max\left(\sqrt{\frac{\cos^2(s)}{2} + \frac{\sin^2(s)}{2}}\right),$$



(a) Ellipse at $t = 1$. (b) Ellipse at $t = 2$. (c) Ellipse at $t = 3$.

(d) Crocked-egg at $t = 1$. (e) Crocked-egg at $t = 2$. (f) Crocked-egg at $t = 3$.

Fig. 3. Deformation of an ellipse and a crocked egg while maintaining their area constant.

then, we obtain the minimum velocity $v_{\min} > 1/\sqrt{2}$.

V. Experiments and Results

As we mentioned in Section II, we propose a general model for any shape function Φ. In our experiments, we consider two different shapes to validate our technique in simulations and with actual robots. The shapes are the ellipse and the crooked egg. Both shapes have some parameters that influence the perimeter and the area. By constraining these parameters, we might maintain the corresponding area constant during deformations to pass through obstacles and achieve the goal. The area is an important constraint because it allows the escorted agents to move within the dynamic shape by reconfiguring their formation.

The deformable behavior plays a vital role when the shape (and correspondingly the robots) must pass though a narrow passageway or a corner. Figures 3(a-c) illustrate how the deformable ellipse can pass through a reduced space and simultaneously maintaining its constant area. Figure 3(d-f) shows how the crocked egg is flexible enough to bend around corners. Hence, we can choose a corresponding shape depending on the environment. In the rest of this section, we formulate the two kinds of shapes, describe the simulations, and the experiments with actual robots. Watching the accompanying video may help the visualization of the robots executing the computed trajectories.

A. Shape model Formulations

We formulate the ellipse and the crooked-egg functions as follows.

1) Ellipse function: An ellipse centered in $[x, y]^T = [q_2, q_3]^T$ and semi-axes $[r_1, r_2]^T = [q_0, q_1]^T$ was described in Eq. (1). In order to maintain a specific area A in this shape, we use the area constraint $A = \pi r_1 r_2$, to define r_2 as a function of r_1 as, $r_2 = \frac{A}{\pi . r_1}$. Then, we obtain the ellipse function with $n = 3$ degrees of freedom,

$$\Phi_e(s, \mathbf{q}) = \begin{bmatrix} q_0\cos(2\pi s) + q_1 \\ \frac{A}{\pi q_0}\sin(2\pi s) + q_2 \end{bmatrix}. \tag{9}$$

We can also add a new degree of freedom $n = 4$ for rotation,

$$\Phi_e(s, \mathbf{q}) = \begin{bmatrix} q_0\cos(2\pi s)\cos(q_3) - \frac{A}{\pi q_0}\sin(2\pi s)\sin(q_3) + q_1 \\ \frac{A}{\pi q_0}\sin(2\pi s) + q_0\sin(2\pi s)\cos(q_3) + q_2 \end{bmatrix}, \tag{10}$$

where the configuration vector is $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$.

2) Crooked egg shape: The crooked egg is a shape that can be deformed to adapt to corner-based environments like mazes. The general form of the crooked egg function in polar coordinates (r, θ) is $r(\theta) = c_0 \sin^3\theta + c_1 \cos^3\theta$, where the coefficients c_0 and c_1 determine the orientation, form, and size of the shape. The total area A is given by,

$$A = \int_0^{2\pi} \frac{r^2(\theta)}{2} d\theta = \frac{5\pi}{16}(c_0^2 + c_1^2).$$

In order to keep the area constant, we can change c_0 as a function of c_1 or vice-versa. If we say that c_0 or c_1 is c then we can write the other as:

$$f(c) = \sqrt{\frac{16A}{5\pi} - c^2}.$$

where $c \leq \sqrt{16A/5\pi}$. We avoided the negative value of f because it gives the orientation of the shape and we will include it in the r function. Then the function r can be extended by adding a parameter $q_0 \in [-4, 4]$ that determines the form and orientation of the crooked-egg shape,

$$r(\theta, q_0) = \begin{cases} (q_0+3)\sin^3\theta + f(q_0+3)\cos^3\theta, & \text{if } q_0 \in [-4,-2) \\ f(q_0+1)\sin^3\theta - (q_0+1)\cos^3\theta, & \text{if } q_0 \in [-2,0) \\ (1-q_0)\sin^3\theta - f(q_0-1)\cos^3\theta, & \text{if } q_0 \in [0,2) \\ -f(q_0-3)\sin^3\theta - (q_0-3)\cos^3\theta, & \text{if } q_0 \in [2,4]. \end{cases}$$

By converting from polar coordinates to Cartesian coordinates, we can obtain the crooked egg model as

$$\Phi(s, \mathbf{q}) = \begin{bmatrix} r(2\pi s, q_0)\cos(2\pi s) + q_1 \\ r(2\pi s, q_0)\sin(2\pi s) + q_2 \end{bmatrix}, \qquad (11)$$

where q_1 and q_2 are the translation factors, and q_0 determines the form and the orientation of the crooked-egg.

B. Simulations

In order to validate our proposed method, we initially created two different scenarios for simulations. The first one uses the ellipse function (Eq. (10)) to navigate in an environment with narrow spaces vertically and diagonally. By using our motion planning algorithm (Alg. 1), the shape is transformed to pass through the obstacles, from a starting point S to a goal point G, by adapting its orientation and its anchor. We use the boundary function $\gamma(t, s)$ to obtain four different robot trajectories $S_i(t)$, $i = 1, ..., 4$. Figure 4 shows an snapshot of the trajectories that each robot follows. The robots are represented by small circles with different colors, and the continuous lines connected to each robot as a tail, represent the movement during the last $660s$ before the snapshot.

The second scenario uses the crooked-egg shape (Eq. (11)) to escort a constant area, from a start point S to a goal point G, in a maze. The robot trajectories are shown in Figure 5, the tail shows the motion of the last $330s$ before the snapshot. It is possible to see how the obtained crooked-egg function adapts its shape, as planned by Algorithm 1, to pass through the corners.

In both scenarios, the robots maintain a constant linear velocity $v = 0.39m/s$ while they move around the dynamic boundary. We can see that the resulting trajectory does not collide with obstacles or walls as we are navigating in the free configuration space.

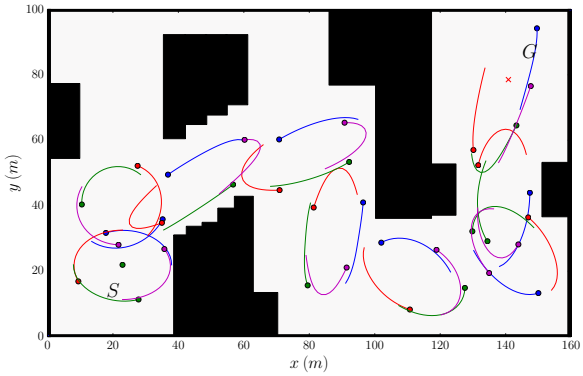


Fig. 4. Escorting an ellipse shape with constant area in an environment with obstacles. Snapshot of four robot trajectories. The small circles represent the robots and the continuous lines are the recent movement before the snapshot.

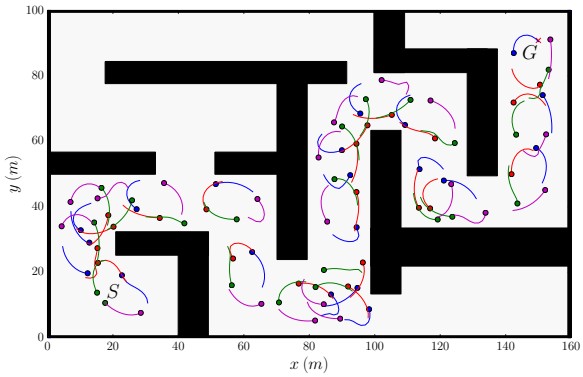


Fig. 5. Escorting a crooked-egg shape in a maz. The walls are represented by the black rectangles.

We found that the robots start in approximately equidistant positions, but depending on the boundary dynamics, the robots have an attractive behavior to the points that have the tangent vector in the same direction of the shape's translation. However this phenomenon is reduced when the robots' velocity v is high in comparison to the shape's translational velocity.

C. Experiments with real robots

To evaluate the applicability of the proposed technique in real world with the presence of external noise and unmodeled dynamics, we have performed some experiments with a team of four e-puck robots (See Fig. 6). Our testbed is composed of an open map with dimensions of $150 \times 125\ cm$ and localization based on a camera mounted on the ceiling. We used the Robot Operating System (ROS) as implementation too. To command each e-puck (differential) robot i, we convert the robot input of Equation (7) to send linear and angular velocities (v_i, ω_i), $i = 1, ..., 4$ based on the static feedback linearization scheme [16] as follows,

$$\begin{bmatrix} v_i \\ w_i \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \frac{-\sin(\theta)}{d} & \frac{\cos(\theta)}{d} \end{bmatrix} \mathbf{w}_i, \qquad (12)$$

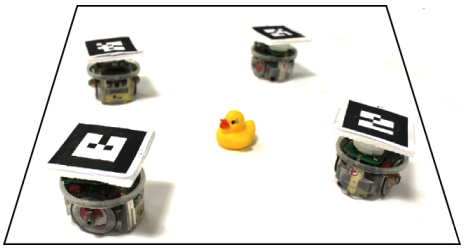
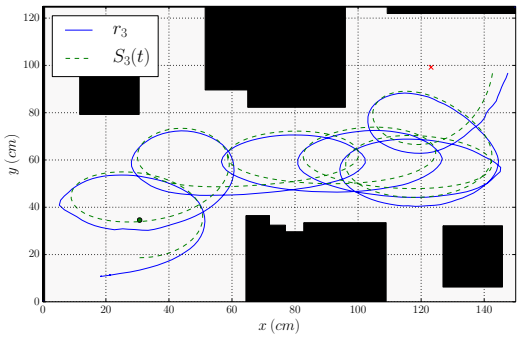
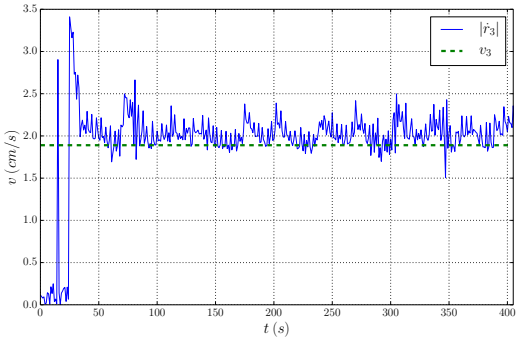


Fig. 6. A team of four E-puck robots.



(a) Trajectory for Robot $\mathbf{r}_3$. The dashed line represents the desired trajectory $S_3(t)$



(b) Linear velocity for Robot 3 and the desired constant velocity v_3.

Fig. 7. Real Robot motion in comparison to the desired motion.

where θ and d are the robot orientation with respect to the global frame and the constant offset point parameter, respectively.

A trial of the real robot experiment and the computed trajectories can be watched in the accompanying video `[https://www.youtube.com/watch?v=Ug5Ne6v34iQ]`. In Figure 7(a), we illustrate how robot 3 follows the trajectory $S_3(t)$. It is possible to see that the robot follows the trajectory very close but not exactly, for this reason, we must take into account a margin of possible collisions to increase the size of the obstacles before the shape motion planning. We can see in Fig. 7(b) how the proportional controller tries to maintain the constant velocity v_3. At the beginning, we get a peak in the velocity until the convergence.

VI. Conclusions and Future Work

In this paper, we proposed a surveillance method to escort a set of agents from one place to a goal. We stated and proposed a solution for two problems for *dynamic perimeter surveillance*. The first one states how to estimate the trajectory of a boundary to guard the agents, by generating a plan to move and to deform a boundary shape. We approached this problem by using a modification of the classical kinodynamic motion planning technique to create an analytical continuous function for the boundary behavior. The second problem states how to move the robots around the boundary function with a desired constant velocity. We proposed a method that takes the boundary function to create the robot trajectories. These trajectories are obtained of resolving a differential equation in each robot distributively. We validated our proposal in simulations and with real robots. In our experiments, we showed the flexibility and effectiveness of our method by testing in different scenarios. With two different shapes, the ellipse and the crooked egg, we can encompass a finite area around complex environments.

As a future work, we would like to extend this method to decentralize the whole planning process. Furthermore, we also would like to change the deformation strategy regarding to application conditions, i.e. using obstacles as an aid to protect the internal agents and distribute the robots along the open area.

References

[1] F. Zhang and N. E. Leonard, "Coordinated patterns of unit speed particles on a closed curve," *Systems & control letters*, vol. 56, no. 6, pp. 397–407, 2007.

[2] N. Ceccarelli, M. Di Marco, A. Garulli, and A. Giannitrapani, "Collective circular motion of multi-vehicle systems," *Automatica*, vol. 44, no. 12, pp. 3025–3035, 2008.

[3] L. Pimenta, G. A. Pereira, M. M. Gonçalves, N. Michael, M. Turpin, and V. Kumar, "Decentralized controllers for perimeter surveillance with teams of aerial robots," *Advanced Robotics*, vol. 27, no. 9, pp. 697–709, 2013.

[4] L. Sabattini, C. Secchi, and C. Fantuzzi, "Closed-curve path tracking for decentralized systems of multiple mobile robots," *Journal of Intelligent & Robotic Systems*, vol. 71, no. 1, pp. 109–123, 2013.

[5] C. Belta and V. Kumar, "Abstraction and control for groups of robots," *Robotics, IEEE Transactions on*, vol. 20, no. 5, pp. 865–875, 2004.

[6] C. C. Cheah, S. P. Hou, and J. J. E. Slotine, "Region-based shape control for a swarm of robots," *Automatica*, vol. 45, no. 10, pp. 2406–2411, 2009.

[7] S. P. Hou and C. C. Cheah, "Dynamic compound shape control of robot swarm," *Control Theory & Applications, IET*, vol. 6, no. 3, pp. 454–460, 2012.

[8] Y. Cao and R. Fierro, "Dynamic boundary tracking using dynamic sensor nets," in *Decision and Control, 2006 45th IEEE Conference on*, Dec 2006, pp. 703–708.

[9] H. M. Choset, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005, ch. 7.

[10] S. M. Lavalle, J. J. Kuffner, and Jr., "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, 2000, pp. 293–308.

[11] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotic Research*, vol. 30, no. 17, pp. 846–894, 2011.

[12] D. Schneider, E. Schomer, and N. Wolpert, "Completely randomized rrtconnect: A case study on 3d rigid body motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[13] L. Palmieri and K. O. Arras, "Distance metric learning for rrt-based motion planning with constant-time inference," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[14] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, pp. 378–400, 2001.

[15] T. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, Sept 2014, pp. 3713–3719.

[16] J. Desai, J. Ostrowski, and V. Kumar, "Controlling formations of multiple mobile robots," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 4, May 1998, pp. 2864–2869 vol.4.