Proceedings of the Twenty-Ninth International
Florida Artificial Intelligence Research Society Conference

Multi-Agent Area Coverage Control Using Reinforcement Learning

Adekunle A. Adepegba[1], **Suruz Miah**[2], and **Davide Spinello**[1]
[1]Department of Mechanical Engineering, University of Ottawa, ON, CANADA
[2] Electrical and Computer Engineering, Bradley University, Peoria, IL, USA

Abstract

An area coverage control law in cooperation with reinforcement learning techniques is proposed for deploying multiple autonomous agents in a two-dimensional planar area. A scalar field characterizes the risk density in the area to be covered yielding nonuniform placement of agents while providing optimal coverage. This problem has traditionally been addressed in the literature to date using conventional control techniques, such as proportional and proportional–derivative controllers. In most cases, agents' actuator energy required to drive them in optimal configurations in the workspace is not taken into considerations. Here the maximum coverage is achieved with minimum actuator energy required by each agent. Similar to existing coverage control techniques, the proposed algorithm takes into consideration time-varying risk density. Area coverage is modeled using Voronoi tessellations governed by agents. Theoretical results are demonstrated through a set of computer simulations where multiple agents are able to deploy themselves, thus paving the way for efficient distributed Voronoi coverage control problems.

Introduction

The problem of cooperative multi-agent decision making and control is to deploy a group of agents over an environment to perform various tasks including sensing, data collection and surveillance. This topic covers a wide range of applications in varied fields. Applications in military and civilian domains, such as harbor protection (Simetti et al. 2010; Kitowski 2012; Miah et al. 2014), perimeter surveillance (Pimenta et al. 2013; Zhang, Fricke, and Garg 2013), search and rescue missions (Hu et al. 2013; Allouche and Boukhtouta 2010), and cooperative estimation (Spinello and Stilwell 2014).

In the last two decades, researchers have proposed various solutions to a lot of interesting sensor network coverage problems based on the work of (Cortes et al. 2002). Recent contributions and meaningful extensions of the framework devised in (Cortes et al. 2004) have been proposed in the literature (Bullo, Cortés, and Martínez 2009; Martinez, Cortes, and Bullo 2007). In (Corts, Martnez, and Bullo 2005) the problem of limited-range interaction between agents was addressed. The work described in this paper is similar the coverage control problem presented in (Lee, Diaz-Mercado, and Egerstedt 2015a; 2015b; Miah et al. 2014) which uses the geometrical notion of a Voronoi partition to assign the agents to different parts of an environment.

Many processes in industry can benefit from control algorithms that learn to optimise a certain cost function. Reinforcement learning (RL) is such a learning method. The user sets a certain goal by specifying a suitable reward function for the RL controller. The RL controller then learns to maximize the cumulative reward received over time in order to reach that goal. The proposed recursive least square actor-critic Neural Network (NN) solution implemented in this paper is preferred over the gradient descent approach (Al-Tamimi, Lewis, and Abu-Khalaf 2008; Vamvoudakis and Lewis 2009; Vrabie et al. 2007; Vrabie and Lewis 2009) since it offers significant advantages including the ability to extract more information from each additional observation (Bradtke, Ydstie, and Barto 1994) and would thus be expected to converge with fewer training samples. Moreover, this paper implements the synchronous discrete-time adaptation of both actor and critic NNs.

The contributions of the paper are two folds. First, it involves the formulation of a nonlinear error coverage function linearized for dynamic discrete-time multi-agent systems, where information flow is restricted by a communication graph structure. Most of the prior works in the Voronoi-based coverage control consider single-integrator dynamics for the agents. However, in practice a wide range of mobile agents such as unmanned vehicles have more complex dynamics, which can invalidate the performance of the algorithms developed for trivial dynamics. Second, we develop an adaptive controller using actor-critic NN approximation that asymptotically drives agents in optimal configuration such that the coverage is maximized regardless of the complexity of the time-varying density throughout the workspace. Like that, the agents employ minimum actuator energy to place themselves in optimal configurations. The rest of this paper is organized as follows. First, Voronoi-based area coverage problem is introduced and formulated. Agent deployment using a time-varying density model is introduced which is a function of position of some unknown targets in the environment. The second order control law is then introduced and the nonlinear error coverage function is formulated. The proposed multi-agent area coverage control

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

(MAACC) law in cooperation with reinforcement learning techniques is then illustrated. Following that, the key steps of the proposed algorithm are summarized. The paper concludes with some numerical simulations followed by conclusion and future research avenues.

System Model and Problem Formulation

We consider a group of n homogeneous agents, where the dynamics of ith, $i = 1, \ldots, n$, agent is modeled by

$$\ddot{\mathbf{p}}_i = \mathbf{u}_i, \tag{1}$$

where agent's position $\mathbf{p}_i = [x_i, y_i]^T$, and its corresponding acceleration vector is denoted by $\mathbf{u}_i \in \mathbb{R}^2$. Agents are deployed in a 2D area $\Omega \subset \mathbb{R}^2$.

Each agent partitions the workspace (or area to be covered) according to Voronoi tessellation technique presented in (Miah et al. 2015), such that $\Omega = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \ldots, \cup \mathcal{V}_n$, where the area ($i$th Voronoi cell) $\mathcal{V}_i \subset \Omega$ belongs to agent i, $i = 1, \ldots, n$. The Voronoi region $\mathcal{V}_i$ of the ith agent is the locus of all points that are closer to it than to any other agents, *i.e.,* $\mathcal{V}_i = \{\mathbf{q} \in \Omega | \parallel \mathbf{q} - \mathbf{p}_i \parallel \leq \parallel \mathbf{q} - \mathbf{p}_j \parallel, i \neq j, i \in 1, 2, ..., n\}$. The risk associated with the agents' workspace is modeled by the time-varying risk density function defined as

$$\phi(\mathbf{q}, t) = \phi_0 + e^{\left\{-\frac{1}{2}\left(\frac{(q_x - \bar{q}_x(t))^2}{\beta_x^2} + \frac{(q_y - \bar{q}_y(t))^2}{\beta_y^2}\right)\right\}}, \tag{2}$$

where $\phi_0 > 0$ is constant and $\bar{\mathbf{q}}(t) = [\bar{q}_x(t), \bar{q}_y(t)]^T$ is the position of a moving target that characterizes risk throughout the workspace Ω. Clearly, the mass and centroid of the ith Voronoi cell are given by $m_{\mathcal{V}_i} = \int_{\mathcal{V}_i} \phi(\mathbf{q}) \mathrm{d}\mathbf{q}$ and $\mathbf{c}_{\mathcal{V}_i} = \frac{1}{m_{\mathcal{V}_i}} \int_{\mathcal{V}_i} \mathbf{q}\phi(\mathbf{q}) \mathrm{d}\mathbf{q}$. Intuitively, for optimal coverage by a team of agents, more (less) agents should be deployed where higher (lower) values of the measure risk density. Furthermore, we assume that the sensing performance function $f(r_i)$ of the ith agent is Lebesgue measurable and that it is strictly decreasing with respect to the Euclidean distance $r_i = \|\mathbf{q} - \mathbf{p}_i\|$. Hence, the sensing performance function of the ith, $i \in \mathcal{I}$, agent is defined as $f(r_i) = a \exp(-br_i^2)$. Motivated by the typical locational optimization problem (Okabe et al. 2000), we define the non-autonomous total coverage metric as

$$H(\mathbf{p}, \mathcal{V}, t) = \sum_{i=1}^{n} \int_{\mathcal{V}_i} f(r_i)\phi(\mathbf{q}, t) \mathrm{d}\mathbf{q}, \tag{3}$$

The model (3) encodes how rich the coverage in Ω is. In other words, the higher H implies that the corresponding distribution of agents achieves better coverage of the area Ω. Hence, the problem can be stated as follows: Given the time-varying density function governed by a moving target (see model (2), we seek to spatially distribute agents such that the coverage H is maximized regardless of the complexity of the risk density ϕ, i.e.,

$$\sup_{\mathbf{p}} H(\mathbf{p}, \mathcal{V}, t), \text{ subject to (1) as } t \to \infty, \tag{4}$$

where $\mathbf{p} \equiv [\mathbf{p}_1^T, \mathbf{p}_2^T, \ldots, \mathbf{p}_n^T]^T$. In the following section, we illustrate multi-agent area coverage control algorithm for solving the problem (4).

Multi-Agent Area Coverage Control Law

A standard setup for solving the problem (4) follows the Lloyd algorithm and is illustrated in (Cortes et al. 2004), where the feedback control law for the ith agent (1) is given by:

$$\mathbf{u}_i(t) = -2K_p m_{\mathcal{V}_i}(\mathbf{p}_i(t) - \mathbf{c}_{\mathcal{V}_i}(t)) - K_d \dot{\mathbf{p}}_i(t), \tag{5}$$

where all agents asymptotically converge to their centroids for $K_p,\ K_d > 0$, given the fact that the density ϕ is time-invariant. Since we considered time-varying density function $\phi(\mathbf{q}, t)$, using (5), and defining $\mathbf{e}_i(t) = \mathbf{c}_{\mathcal{V}_i}(t) - \mathbf{p}_i(t)$, we define the error dynamical model as

$$\dot{\mathbf{e}}_i = \boldsymbol{\eta}(\mathbf{e}_i, \mathbf{u}_i) = \frac{2K_p m_{\mathcal{V}_i}}{K_d}\left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right)\mathbf{e}_i + \frac{1}{K_d}\left(\mathbf{I}_2 - \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right)\mathbf{u}_i + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t}. \tag{6}$$

Model (6) represents the continuous time nonlinear error dynamics for agent i, where the vector-valued vector function $\boldsymbol{\eta}$:$\mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^2$ evolves the error $\mathbf{e}$ as time $t \to \infty$. Hence, the corresponding linear model which defines the systems dynamic behaviour about the optimal equilibrium operating point $(\mathbf{0}, \mathbf{0})$ is given by

$$\dot{\mathbf{e}}_i = \mathbf{A}\mathbf{e}_i + \mathbf{B}\mathbf{u}_i, \tag{7}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are defined as

$$\mathbf{A} = \frac{2K_p m_{\mathcal{V}_i}}{K_d}\left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right), \ \mathbf{B} = \frac{1}{K_d}\left(\mathbf{I}_2 - \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right).$$

Since the control law will be implemented in a digital computer, let $t = kT$, $k = 0, 1, \ldots,$ and T is the sampling time. The discrete time model of (7) can be written as

$$\mathbf{e}_i(k+1) = \mathbf{A}^d \mathbf{e}_i(k) + \mathbf{B}^d \mathbf{u}_i(k) \tag{8}$$

where $\mathbf{A}^d = \exp(\mathbf{A}T)$ and $\mathbf{B}^d = \int_0^T \exp(\mathbf{A}T) \mathrm{d}\tau \mathbf{B}$. Model (8) can be represented in the following form:

$$\mathbf{e}_i(k+1) = \mathbf{f}(\mathbf{e}_i(k)) + \mathbf{g}(\mathbf{e}_i(k))\mathbf{u}_i(k), \tag{9}$$

The 2D position and velocity of the ith agent at time instant k can be described by the discrete time dynamical model :

$$\begin{bmatrix} \mathbf{p}_i(k+1) \\ \mathbf{v}_i(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_i(k) \\ \mathbf{v}_i(k) \end{bmatrix} + \begin{bmatrix} (1/2)T^2 & 0 \\ 0 & (1/2)T^2 \\ T & 0 \\ 0 & T \end{bmatrix} \mathbf{u}_i(k) \tag{10}$$

where $\mathbf{p}_i(k) = [x_i(k), y_i(k)]^T$ is the ith agent's position, $\mathbf{v}_i(k) = [v_i^x(k), v_i^y(k)]^T$ is its velocity, and $\mathbf{u}_i(k) = [u_i^x(k), u_i^y(k)]^T$ is its 2D acceleration inputs at time instant k.

Using (LaSalle 1960) invariance principle it can be proved that the control input (5) converges to a centroidal Voronoi configuration and provides a locally optimal coverage over the region (Cortes et al. 2004). According to (Lloyd 2006), the control strategy in which agent moves towards the centroid of its Voronoi cell locally solves the area coverage control problem. We propose an actor-critic NN approximation method used to approximate the control law (5) that can be applied to more general coverage control problems, where the density function is not explicitly known a priori. In addition, actor-critic NN approximation of the control law (5) incorporates minimum actuating energy applied to agents' actuators.

Actor-Critic Reinforcement Learning

We consider the linear system (9) where $\mathbf{e}_i \in \mathbb{R}^2$,$\mathbf{f}(\mathbf{e}_i) \in \mathbb{R}^2$, $\mathbf{g}(\mathbf{e}_i) \in \mathbb{R}^{2\times 2}$ with control input $\mathbf{u}_i \in \mathbb{R}^2$. There exist a control input $\mathbf{u}_i(k)$ that minimizes the Value function given as

$$V(\mathbf{e}_i(k)) = \sum_{\kappa=k}^{\infty} \left(\mathbf{e}_i^T(\kappa)\mathbf{Q}\mathbf{e}_i(\kappa) + \mathbf{u}_i^T(\kappa)\mathbf{R}\mathbf{u}_i(\kappa)\right) \quad (11)$$

where $\mathbf{Q} \in \mathbb{R}^{2\times 2}$ and $\mathbf{R} \in \mathbb{R}^{2\times 2}$ are positive definite matrices i.e. $\mathbf{e}_i^T(k)\mathbf{Q}\mathbf{e}_i(k) > 0, \forall \mathbf{e}_i \neq 0$ and $\mathbf{e}_i^T(k)\mathbf{Q}\mathbf{e}_i(k) = 0$ when $\mathbf{e}_i = 0$. Note the second term of the value function (11). It takes into account the asymptotic energy consumption of the agents.

Equation (11) can be rewritten in the form

$$\begin{aligned} V(\mathbf{e}_i(k)) &= \mathbf{e}_i^T(k)\mathbf{Q}\mathbf{e}_i(k) + \mathbf{u}_i^T(k)\mathbf{R}\mathbf{u}_i(k) \\ &+ \sum_{\kappa=k+1}^{\infty} \left(\mathbf{e}_i^T(\kappa)\mathbf{Q}\mathbf{e}_i(\kappa) + \mathbf{u}_i^T(\kappa)\mathbf{R}\mathbf{u}_i(\kappa)\right) \Rightarrow \\ V(\mathbf{e}_i(k)) &= \mathbf{e}_i^T(k)\mathbf{Q}\mathbf{e}_i(k) + \mathbf{u}_i^T(k)\mathbf{R}\mathbf{u}_i(k) + \\ &\qquad V(\mathbf{e}_i(k+1)). \end{aligned} \quad (12)$$

Hence, we find the control inputs $\mathbf{u}_i$ such that the value function (11) is minimized, *i.e.,*

$$\begin{aligned} V^*(\mathbf{e}_i(k)) &= \min_{\mathbf{u}_i(k)} \\ &\left[\mathbf{e}_i(k)^T\mathbf{Q}\mathbf{e}_i(k) + \mathbf{u}_i(k)^T\mathbf{Q}\mathbf{u}_i(k) + V^*(\mathbf{e}_i(k+1))\right] \end{aligned} \quad (13)$$

Model (13) is the discrete-time Hamilton-Jaccobi-Bellman equation. The optimal control, $\mathbf{u}_i^*$ can be obtained by finding the gradient of the right hand side of (13) and setting it to zero i.e. $\left(\frac{\partial V^*(\mathbf{e}_i(k))}{\partial \mathbf{u}_i} = 0\right)$, we obtain

$$2\mathbf{u}_i\mathbf{R} + \mathbf{g}^T(\mathbf{e}_i(k))\frac{\partial V^*(\mathbf{e}_i(k+1))}{\partial \mathbf{e}_i(k+1)} = 0 \quad (14)$$

$$\mathbf{u}_i^*(\mathbf{e}_i(k)) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{g}^T(\mathbf{e}_i(k))\frac{\partial V^*(\mathbf{e}_i(k+1))}{\partial \mathbf{e}_i(k+1)} \quad (15)$$

$V^*(\mathbf{e}_i(k))$ represents the value function consistent with the optimal control policy $\mathbf{u}_i^*(\mathbf{e}_i(k))$. Since the HJB equation is a nonlinear equation, it is generally difficult or impossible to obtain its solution. Therefore we propose an actor critic NN approximation RL algorithm that approximates both the value function and the control policy.

Select a value function approximation and control action approximation structure as

$$\hat{V}_j(\mathbf{e}_i(k)) = \mathbf{w}_{c,j}^T\boldsymbol{\rho}(\mathbf{e}_i(k)) \quad (16)$$

$$\hat{\mathbf{u}}_j(\mathbf{e}_i(k)) = \mathbf{W}_{a,j}^T\boldsymbol{\sigma}(\mathbf{e}_i(k)) \quad (17)$$

respectively, where $\hat{V}$ and $\hat{\mathbf{u}}$ are estimates of the value function and control action respectively, $\mathbf{w}_{c,j} \in \mathbb{R}^{N_c}$ is the weight vector of critic NN, $\mathbf{W}_{a,j} \in \mathbb{R}^{N_a\times 2}$ is the weight matrix of actor NN, N_c and N_a are the number of neurons in the critic and actor NN respectively, j is the iteration step for both actor and critic NN, $\boldsymbol{\rho}(\mathbf{e}_i(k))$ and $\boldsymbol{\sigma}(\mathbf{e}_i(k))$ are the activation functions of the critic and actor NN respectively. The target value function and control function can be defined as

$$\begin{aligned} V_{j+1}(\mathbf{e}_i(k)) &= \mathbf{e}_i(k)^T\mathbf{Q}\mathbf{e}_i(k) + \mathbf{u}_i(k)^T\mathbf{R}\mathbf{u}_i(k) + \\ V_j(\mathbf{e}_i(k+1)) &= \mathbf{e}_i(k)^T\mathbf{Q}\mathbf{e}_i(k) + \mathbf{u}_i(k)^T\mathbf{R}\mathbf{u}_k + \\ &\qquad \mathbf{w}_{c,j}^T\boldsymbol{\rho}(\mathbf{e}_i(k+1)) \end{aligned} \quad (18)$$

$$\begin{aligned} &\mathbf{u}_{j+1}(\mathbf{e}_i(k)) = \\ &\qquad -\frac{1}{2}\mathbf{R}^{-1}\mathbf{g}^T(\mathbf{e}_i(k))\nabla\boldsymbol{\rho}^T(\mathbf{e}_i(k+1))\mathbf{w}_{c,j} \end{aligned} \quad (19)$$

where V_{j+1} and $\mathbf{u}_{j+1}$ can be defined as the target value function and control action respectively.

The generalized temporal difference error equation for the critic NN can be derived as:

$$\delta_{c,j}(k) = \mathbf{w}_{c,j+1}^T\boldsymbol{\rho}(\mathbf{e}_i(k)) - V_{j+1}(\mathbf{e}_i(k)). \quad (20)$$

We define the critic NN least squares error $E_{c,j}^2$ to be minimized as

$$\begin{aligned} E_{c,j}^2 &= \sum_{l=1}^{N_t}\frac{1}{2}\left(\delta_{c,j}^l(k)\right)^2 \\ &= \sum_{l=1}^{N_t}\frac{1}{2}\left[\mathbf{w}_{c,j+1}^T\boldsymbol{\rho}^l(\mathbf{e}_i(k)) - V_{j+1}^l(\mathbf{e}_i(k))\right]^2 \end{aligned} \quad (21)$$

where ($l = 1, \ldots, N_t$) and N_t=Number of training steps. Differentiating $E_{c,j}^2$ with respect to $\mathbf{w}_c$, we get

$$\begin{aligned} &\frac{\partial E_{c,j}^2}{\partial \mathbf{w}_{c,j+1}} = \left(\frac{\partial E_{c,j}^2}{\partial \delta_{c,j}}\right)\left(\frac{\partial \delta_{c,j}}{\partial \mathbf{w}_{c,j+1}}\right) \\ &= \frac{1}{2}\sum_{l=1}^{N_t}\left[2\left(\mathbf{w}_{c,j+1}^T\boldsymbol{\rho}^l(\mathbf{e}_i(k)) - V_{j+1}^l(\mathbf{e}_i(k))\right)\right]\boldsymbol{\rho}^l(\mathbf{e}_i(k)). \end{aligned} \quad (22)$$

Equating (22) to zero and rearranging the equation, we obtain the critic weight update least square solution of the form:

$$\begin{aligned} \hat{\mathbf{w}}_{c,j+1} &= \left(\sum_{l=1}^{N_t}\boldsymbol{\rho}^l(\mathbf{e}_i(k))(\boldsymbol{\rho}^l(\mathbf{e}_i(k)))^T\right)^{-1} \\ &\qquad \left(\sum_{l=1}^{N_t}\boldsymbol{\rho}^l(\mathbf{e}_i(k))V_{j+1}^l(\mathbf{e}_i(k))\right) \end{aligned} \quad (23)$$

Likewise, we define the actor NN training error as

$$\boldsymbol{\delta}_{a,j}(k) = \mathbf{W}_{a,j+1}^T \boldsymbol{\sigma}(\mathbf{e}_i(k)) - \mathbf{u}_{j+1}(\mathbf{e}_i(k)) \quad (24)$$

where $\boldsymbol{\delta}_{a,j}$ is the actor training error, we define the actor NN LS error $E_{a,j}^2$ to be minimized as

$$\begin{aligned} E_{a,j}^2 &= \sum_{l=1}^{N_t} \frac{1}{2} \left(\boldsymbol{\delta}_{a,j}^l(k)\right)^2 \\ &= \frac{1}{2} \sum_{l=1}^{N_t} \left[\mathbf{W}_{a,j+1}^T \boldsymbol{\sigma}^l(\mathbf{e}_i(k)) - \mathbf{u}_{j+1}^l(\mathbf{e}_i(k))\right]^2 \quad (25) \end{aligned}$$

Differentiating $E_{a,j}^2$ in terms of $\mathbf{W}_a$, we obtain

$$\frac{\partial E_{a,j}^2}{\partial \mathbf{W}_{a,j+1}} = \left(\frac{\partial E_{a,j}^2}{\partial \boldsymbol{\delta}_{a,j}}\right)\left(\frac{\partial \boldsymbol{\delta}_{a,j}}{\partial \mathbf{W}_{a,j+1}}\right) = \frac{1}{2}\sum_{l=1}^{N_t} \left(2[\mathbf{W}_{a,j+1}^T \boldsymbol{\sigma}^l(\mathbf{e}_i(k)) - \mathbf{u}_{j+1}^l(\mathbf{e}_i(k))]\right) \boldsymbol{\sigma}^l(\mathbf{e}_i(k)). \quad (26)$$

Equating (26) to zero and rearranging the equation, we obtain actor NN training target function as follows:

$$\begin{aligned} \mathbf{u}_{j+1}(\mathbf{e}_i(k)) &= \hat{\mathbf{W}}_{a,j}^T \boldsymbol{\sigma}(\mathbf{e}_i(k)) \\ &= -\frac{1}{2}\mathbf{R}^{-1}\mathbf{g}^T(\mathbf{e}_i(k))\nabla\boldsymbol{\rho}^T(\mathbf{e}_i(k+1))\hat{\mathbf{w}}_{c,j} \quad (27) \end{aligned}$$

while the actor weights least square solution is of the form:

$$\hat{\mathbf{W}}_{a,j+1} = \left(\sum_{l=1}^{N_t} \boldsymbol{\sigma}^l(\mathbf{e}_i(k))(\boldsymbol{\sigma}^l(\mathbf{e}_i(k)))^T\right)^{-1} \left(\sum_{l=1}^{N_t} \boldsymbol{\sigma}^l(\mathbf{e}_i(k))\mathbf{u}_{j+1}^l(\mathbf{e}_i(k))\right) \quad (28)$$

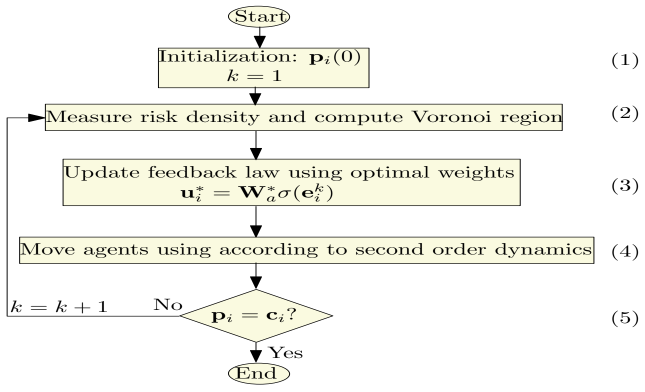
The least square solution weights $\hat{\mathbf{w}}_{c,j}$ and $\hat{\mathbf{W}}_{a,j}$ guarantee system stability as well as convergence to the optimal value and control (Vamvoudakis and Lewis 2009). The actor-critic NN weights are synchronized as follows:

$$\hat{\mathbf{W}}_{a,j+1}(k) = -[\alpha\left(\hat{\mathbf{W}}_{a,j+1}(k) - \hat{\mathbf{w}}_{c,j+1}(k)\right) + (1-\alpha)\hat{\mathbf{w}}_{c,j+1}(k)] \quad (29)$$

where α is the learning rate for the actor-critic NN. Note that for the inverse of matrix of the critic $\left(\boldsymbol{\rho}(\mathbf{e}_i(k))\boldsymbol{\rho}^T(\mathbf{e}_i(k))\right)^{-1}$ and actor $\left(\boldsymbol{\sigma}(\mathbf{e}_i(k))\boldsymbol{\sigma}^T(\mathbf{e}_i(k))\right)^{-1}$ to exist, one needs the basis functions $\boldsymbol{\rho}^T(\mathbf{e}_i(k))$ and $\boldsymbol{\sigma}^T(\mathbf{e}_i(k))$ to be linearly independent and the number of random states to be greater than or equal to the number of neurons, N_c and N_a for the critic and actor respectively (i.e. the matrix determinants must not be zero).

The output of the critic is used in the training process of the actor so that the least square control policy can be computed recursively. In the proposed least square framework, the learning process employs a weight-in-line actor and critic NNs implemented using an recursive least square algorithm to approximate both the actor and critic weights which are tuned synchronously using (28). At each iteration step, the least square solver collects the data needed to calculate states $(\mathbf{e}_i(k))$, and control update $\mathbf{u}_{j+1}(\mathbf{e}_i(k))$ and then finds the weight vectors $\mathbf{W}_{a,j}$ and $\mathbf{w}_{c,j}$ satisfying (23) and (28) respectively, both of which are transferred to the corresponding actor and critic NNs to update their weights. The actor NN generates the control input $\hat{\mathbf{u}}_{j+1}(\mathbf{e}_i(k))$ while the critic NN generates the value function output $\hat{V}_{j+1}(\mathbf{e}_i(k))$.

The actor least square solver generates the optimal action weights $(\hat{\mathbf{W}}_{a,j+1})$, when $\|\hat{V}_{j+1}(\mathbf{e}_i(k)) - \hat{V}_j(\mathbf{e}_i(k))\| \leq \epsilon$, $\|\hat{\mathbf{u}}_{j+1}(\mathbf{e}_i(k)) - \hat{\mathbf{u}}_j(\mathbf{e}_i(k))\| \leq \epsilon$ and $\|\hat{\mathbf{W}}_{a,j+1}(k) - \hat{\mathbf{W}}_{a,j}(k)\| \leq \epsilon$ which is then used in generating the optimal control policy.



Start
Initialization: $\mathbf{p}_i(0)$ $k=1$
(1)
Measure risk density and compute Voronoi region
(2)
Update feedback law using optimal weights $\mathbf{u}_i^* = \mathbf{W}_a^*\sigma(\mathbf{e}_i^k)$
(3)
Move agents using according to second order dynamics
(4)
$k=k+1$
No
$\mathbf{p}_i = \mathbf{c}_i$?
(5)
Yes
End

Figure 1: High level steps (flowchart) of the proposed MAACC algorithm.

MAACC algorithm

The flowchart of the proposed MAACC RL algorithm that converges the agents to their respective centroidal Voronoi configuration is described and shown below:

Step 1: Initialize all parameters such as the initial time and agent's initial position and define the bounds for the workspace.
Step 2: Measure the Risk density ϕ and compute the Voronoi region $\mathcal{V}_i$ obtaining the $m_{\mathcal{V}_i}$, $\mathbf{c}_{\mathcal{V}_i}$, $\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}$, $\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t}$ and $\frac{\partial m_{\mathcal{V}_i}}{\partial t}$.
Step 3: Update the feedback law using optimal weights obtained from actor-critic NN approximation according to (17)
Step 4: The ith agent's new position and velocity are computed using (10)
Step 5: If agent $\mathbf{p}_i$ converges to its centroid $\mathbf{c}_i$ and ϕ is constant, **stop procedure**, else go to **Step 2**

Computer Simulations

To demonstrate the effectiveness of the proposed MAACC algorithm, a MATLAB simulation is conducted on a group of five agents using different scenarios. In all simulations, agents are placed on a 2D convex workspace $\Omega \subset \mathbb{R}^2$ with its boundary vertices at (1.0,0.05), (2.2,0.05), (3.0,0.5), (3.0,2.4), (2.5,3.0), (1.2,3.0), (0.05,2.40), and (0.05,0.4) m. Agents' initial positions are at (0.20,2.20), (0.80,1.78), (0.70,1.35), (0.50,0.93), and (0.30,0.50) m. The sampling

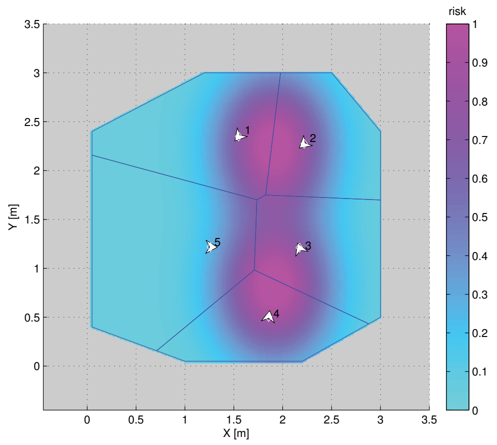


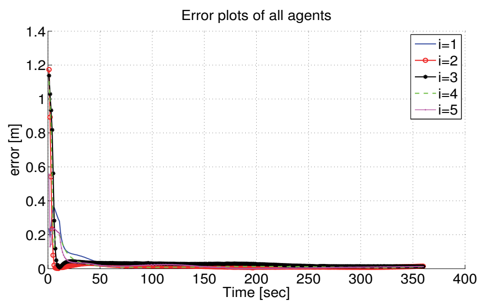
Figure 2: Configurations of five agents at time t=160 s.

time is chosen to be 1 s and the simulation is conducted for 360 s. A moving target inside the workspace characterizes the time-varying risk density with $\beta_x = 0.4$ and $\beta_y = 0.5$. The critic NN and actor NN activation functions are defined as $\boldsymbol{\rho}(\mathbf{e}_i(k)) = [e_1, e_2, e_1^2, 2e_1e_2, e_2^2]$ and $\boldsymbol{\sigma}(\mathbf{e}_i(k)) = [e_1,\ e_2, e_1^2, 2e_1e_2, e_2^2]$, respectively with $N_a = N_c = 5$ and the convergence tolerance $\epsilon = 0.00001$. The actor-critic learning rate $\alpha = 0.0009$, the actor and critic weights are initialized as a zero vector and matrix respectively. The control gains are defined as $K_p = 0.003$ and $K_d = 0.3$, while the sensing performance parameters, $a = 1.0$ and $b = 0.5$.

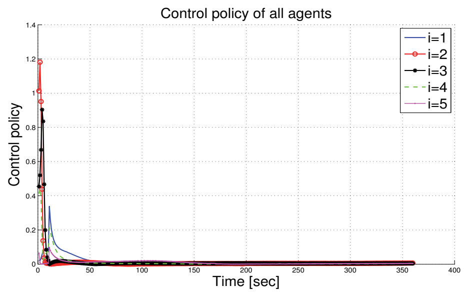
Figure 2 shows the agents configuration at time $t = 160$ s, where agents spatially distribute themselves in an optimal fashion to provide maximum coverage of the area and neutralize the effect of two targets entering into the workspace from two different positions $(2.2, 3.0)$ and $(2.2, 0.05)$. The distance between the agents and their corresponding centroids are shown in Figure 3(a) with their control efforts (speeds) shown in Figure 3(b). These results also show how fast the agents move to their centroids using minimum control inputs (speeds) while providing maximum coverage of the area. The results obtained using the proposed MAACC algorithm are compared to those obtained using the second order control law proposed in (Cortes et al. 2004) which we refer to as CORTES in this experiment. Figure 4(a) shows that proposed algorithm gives a better coverage curve when compared to CORTES, the total coverage cost calculated numerically using (3) is obtained as 2.16×10^6 for the MAACC algorithm which is slightly higher than 2.15×10^6 using CORTES, however, the value function which encodes the control inputs and the error between agents and their centroids obtained by (11) is compared as shown in Figure 4(b), the MAACC algorithm converges faster than CORTES showing minimum energy used.

Summary and Conclusion

We proposed an area coverage control law in cooperation with reinforcement learning techniques, where multiple au-

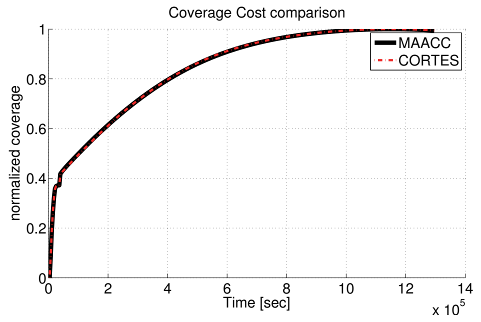


(a)

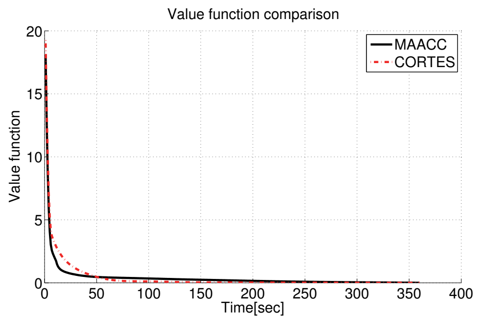


(b)

Figure 3: Performance of the proposed coverage control algorithm(a) Error (Euclidean distance between each agent and its corresponding centroid and (b) agents' control inputs.



(a)



(b)

Figure 4: Proposed algorithm vs Cortes's algorithm (a) coverage metric and (b) value function output.

tonomous agents are able to asymptotically converge themselves in optimal configurations while providing maximum coverage of their 2D workspace. The workspace is partitioned using the well-known Voronoi tessellation technique. Even though a pre-defined time-varying density is considered but the proposed area coverage algorithm is able to solve coverage control problem regardless of the complexity of workspace risk density. An actor-critic NN approximation method is developed as a reinforcement learning technique. In addition, it was shown by simulation that this novel method is able to drive all agents in optimal configurations while minimizing actuator energy required by each agent. The advantage of RL in control theory is that it adjust to changes in environment and it continuously retrain to improve its performance all the time. Additionally, simulations validating the proposed algorithms indeed exhibit the desired behaviours in the virtual environment as well as in theory. A potential future research avenue of the current work is to develop coverage control methods coupled with state estimation of each agents. Like that, agents will be able to deploy themselves in the presence of stochastically intermittent communication network, which is not considered in this manuscript.

References

Al-Tamimi, A.; Lewis, F. L.; and Abu-Khalaf, M. 2008. Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof. *Trans. Sys. Man Cyber. Part B* 38(4):943–949.

Allouche, M. K., and Boukhtouta, A. 2010. Multi-agent coordination by temporal plan fusion: Application to combat search and rescue. *Information Fusion* 11(3):220 – 232. Agent-Based Information Fusion.

Bradtke, S. J.; Ydstie, B. E.; and Barto, A. G. 1994. Adaptive linear quadratic control using policy iteration. Technical report, University of Massachusetts, Amherst, MA, USA.

Bullo, F.; Cortés, J.; and Martínez, S. 2009. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press. Electronically available at http://coordinationbook.info.

Cortes, J.; Martinez, S.; Karatas, T.; and Bullo, F. 2002. Coverage control for mobile sensing networks. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, 1327–1332. IEEE.

Cortes, J.; Martinez, S.; Karatas, T.; and Bullo, F. 2004. Coverage control for mobile sensing networks. *Robotics and Automation, IEEE Transactions on* 20(2):243–255.

Corts, J.; Martnez, S.; and Bullo, F. 2005. Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM: Control, Optimisation and Calculus of Variations* 11:691–719.

Hu, J.; Xie, L.; Lum, K.-Y.; and Xu, J. 2013. Multiagent information fusion and cooperative control in target search. *Control Systems Technology, IEEE Transactions on* 21(4):1223–1235.

Kitowski, Z. 2012. Architecture of the control system of an unmanned surface vehicle in the process of harbour protection. In *Solid State Phenomena*, volume 180, 20–26. Trans Tech Publ.

LaSalle, J. 1960. Some extensions of liapunov's second method. *Circuit Theory, IRE Transactions on* 7(4):520–527.

Lee, S.; Diaz-Mercado, Y.; and Egerstedt, M. 2015a. Multirobot control using time-varying density functions. *IEEE Transactions on Robotics* 31(2):489–493.

Lee, S.; Diaz-Mercado, Y.; and Egerstedt, M. 2015b. Multirobot control using time-varying density functions. *Robotics, IEEE Transactions on* 31(2):489–493.

Lloyd, S. 2006. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.* 28(2):129–137.

Martinez, S.; Cortes, J.; and Bullo, F. 2007. Motion coordination with distributed information. *Control Systems, IEEE* 27(4):75–88.

Miah, S.; Nguyen, B.; Bourque, F.-A.; and Spinello, D. 2014. Nonuniform deployment of autonomous agents in harbor-like environments. *Unmanned Systems* 02(04):377–389.

Miah, M. S.; Nguyen, B.; Bourque, A.; and Spinello, D. 2015. Nonuniform coverage control with stochastic intermittent communication. *IEEE Transactions on Automatic Control* 60(7):1981–1986.

Okabe, A.; Boots, B.; Sugihara, K.; and Chiu, S. N. 2000. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, LTM, second edition.

Pimenta, L. C.; Pereira, G. A.; Gonalves, M. M.; Michael, N.; Turpin, M.; and Kumar, V. 2013. Decentralized controllers for perimeter surveillance with teams of aerial robots. *Advanced Robotics* 27(9):697–709.

Simetti, E.; Turetta, A.; Casalino, G.; and Cresta, M. 2010. Towards the use of a team of usvs for civilian harbour protection: The problem of intercepting detected menaces. In *OCEANS 2010 IEEE - Sydney*, 1–7.

Spinello, D., and Stilwell, D. J. 2014. Distributed full-state observers with limited communication and application to cooperative target localization. *Journal of Dynamic Systems, Measurement, and Control* 136(3):031022.

Vamvoudakis, K., and Lewis, F. 2009. Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, 3180–3187.

Vrabie, D., and Lewis, F. 2009. Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Networks* 22(3):237 – 246. Goal-Directed Neural Systems.

Vrabie, D.; Abu-Khalaf, M.; Lewis, F.; and Wang, Y. 2007. Continuous-time adp for linear systems with partially unknown dynamics. In *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on*, 247–253.

Zhang, G.; Fricke, G.; and Garg, D. 2013. Spill detection and perimeter surveillance via distributed swarming agents. *Mechatronics, IEEE/ASME Transactions on* 18(1):121–129.