An Event-Triggered Virtual Force Algorithm for Multi-Agent Coverage Control with Obstacles

Mohanad Ajina Cameron Nowzari

Abstract—In this work, we propose an event-triggered algorithm based on a virtual force deployment approach to address the multi-agent coverage control problem in the presence of obstacles. Unlike most works that consider this problem, we are mainly interested in reducing the amount of communication and motion required by the agents to reach a configuration that increases the coverage throughout an environment of interest. In particular, most works that consider this problem assume agents are in constant communication with each other. Instead, the event-triggered algorithm we propose allows agents to decide for themselves when communication is necessary while still achieving the primary goal of covering the environment and ensuring collisions are avoided. Several simulations illustrate the result of our algorithm with and without the presence of an obstacle and compares it against a similar algorithm that does not consider event-triggered communication.

I. INTRODUCTION

The topic of wireless sensor networks (WSN) is receiving close attention in many research areas today with applications like sensors monitoring for temperature, traffic, health status, and many others [1], [2], [3], [4]. The number of sensors in the WSN varies depending on the application. The WSN can be built with only a few sensors such as home security WSN [5] or several thousand such as environmental monitoring WSN [6]. As the size of the networks and number of possible applications increase, researchers have been working to improve performance of WSN and to reduce their costs. To address these challenges, efforts have been made to maximize sensor area coverage while reducing the amount of communication between sensors. In this paper, we refer to sensors as agents, and to WSN as a multi-agent system. Our work studies the trade-offs of the multi-agent system performing an area coverage control task when the communication between agents is controlled by an event-triggered law. To achieve this goal, we designed an event-triggered law that allows agents to decide autonomously when updated information about neighbors’ locations is needed to complete the task. Our motivation comes from the need to reduce the performance cost by reducing unnecessary communication.

Literature review: There are two common strategies that have been utilized to address the area coverage control problem. These strategies are computational geometry based and force based [7]. An example of the computational geometry based strategy is a Voronoi diagram or a Voronoi partition. This approach divides an environment into coverage regions based on the distance between agents; more details can be found in [8]. The authors in [9] successfully applied the Voronoi diagram strategy to achieve optimal deployment for mobile sensing networks. However, they assumed that each agent has access to all the agents’ locations at all times. In [10], [11], the assumption was removed, and the Voronoi diagram was implemented based on local information provided by the agent’s neighbors. In [12], the Voronoi diagram with the theoretical techniques was used to calculate the best and worst coverage cases. The above presented Voronoi diagram work assumed free environment. The authors in [13] proposed extension to the Voronoi diagram strategy that addressed the issue of area coverage with heterogeneous agents, and they generalized their approach for non-convex environment. In [14], the authors developed a new approach for area coverage in the presence of known obstacles of non-convex polygonal environments. In [15], obstacles are unknown, and the Voronoi diagram strategy is applied to maximize area coverage.

The second common strategy is force based, that allows agents to move based on attacked attractive and/or repulsive forces. An early work of the force based strategy is proposed in [16] that ensures collision avoidance. In [17], the authors used a repulsive force algorithm to deploy agents in a building. In [18], a target involved virtual force algorithm is proposed to locate more agents closer to an area of interest and to keep agents far away from obstacles. In [19], a virtual force algorithm (VFA) is proposed to increase the coverage area after initially placing agents randomly in the environment in the presence of obstacles. However, they assumed that each agent has access to all agent’s locations at all times. The authors in [20] eliminate this assumption, and they proposed an improved VFA (IVFA) and exponential VFA (VFA). Both algorithms performed better area coverage task than VFA even when agents have limited communication ranges, but on paper, no results considered the presence of obstacles. Recently, in [21], the obstacle avoidance virtual force algorithm (OAVFA) proposed to maximize the area coverage and to minimize the average moving distance. The OAVFA performed better over IVFA and EVFA with and without the presence of an obstacle. However, all the above work assume continuous or periodic communication and/or continuous or periodic sensing among agents.

This brings us to the other area of relevance to this work which are distributed event- and self-triggered coordination strategies. Both types of triggered laws have been proposed to reduce the amount of communication between agents while maintaining some desired system level properties. The
authors in [22] propose a self-triggered algorithm to save communication power for the optimal deployment problem without obstacles. In [23], an event-triggered algorithm was able to reduce the amount of communication between agents when performing the multi-agent rendezvous task. In [24], the proposed event-triggered algorithm saved a significant communication power for the average consensus problem. In [25] and [26], the event-triggered control lowered the number of communication between agents for a leader-following consensus problem, and in [27], for a multi-agent systems consensus problem.

Thus, we are interested in combining the distributed triggering strategies with a force based coverage control strategy in order to improve the performance of the system in terms of reduced communication. More specifically, our work builds on the obstacle avoidance virtual force algorithm (OA VFA) proposed in [21], where the agents are always aware of their neighbors’ positions. Instead, we are interested in combining this motion control algorithm with an event-triggered communication strategy to reduce communication while achieving the same level of coverage as the OA VFA that requires perfect information at all times.

Statement of contributions: The main contribution of our work is the design of the event-triggered virtual force algorithm that reduces communication while agents still complete the primary coverage control task. We first design a motion control law that allows agents to determine their control inputs based on a virtual force deployment approach. Second, we design a decision control law that allows agents to determine when communication with neighbors is needed to complete the task. The event-triggered virtual force algorithm combines both laws to allow agents to deploy in the environment with less communication performed. Our algorithm does not require periodic communication as in [21] while still achieving the same level of coverage. Various simulations illustrate the performance of the event-triggered virtual force algorithm with and without the presence of obstacles.

II. PROBLEM STATEMENT

Consider a network of $n$ agents moving in a rectangle environment $S \in \mathbb{R}^2$ with some static obstacles $O \subset S$. More specifically, we consider $N_o$ distinct obstacles $o_1, \ldots, o_{N_o}$ such that $\bigcup_{m \in \{1, \ldots, N_o\}} o_m = O$. We denote $OS \subset S$ to be the set of environments’ boundaries, and denote the position of agent $i \in \{1, \ldots, n\}$ at discrete time $t \in \mathbb{Z}_{\geq 0}$ to be $p_i^t$. The collection of all agent positions at time $t$ is then given by $P_t = (p_1^t, \ldots, p_n^t) \subset S^n$.

We consider a simple kinematic model with bounded velocity $V_{\text{max}}$.

$$p_{i+1}^t = p_i^t + u_i,$$

where $\|u_i^t\| \leq V_{\text{max}} \Delta t$ is the control input of agent $i$ at time $t$ and $\Delta t$ is the actual time between two discrete time steps.

We assume that the agents are initially unaware of the positions and number of obstacles in the environment $S$. Instead, the agents are able to sense obstacles up to a distance $R_S$ away. We also assume the agents are only able to communicate with neighbors that are agents $R_C$ distance away, $R_C = 2R_S$.

The goal of the agents is now to reach a configuration to cover as much of the unoccupied environment $S \setminus O$ as possible, while keeping the total moving distance and communication among agents as low as possible. More specifically, we consider a binary disk sensing model (BSM) due to its simplicity and effectiveness in modeling covered area [21], [28], [29]. Given the position of an agent $p_i^t$, we say that an arbitrary point in the environment $c \in S$ is covered by agent $i$ at time $t$ if it is within the sensing range, i.e., $\|p_i^t - c\| \leq R_S$. Formally, we define the indicator function

$$\text{BSM}(p_i^t, c) = \begin{cases} 1, & \text{if } \|p_i^t - c\| \leq R_S \\ 0, & \text{otherwise}, \end{cases}$$

which returns 1 if the point $c$ is covered by the agent at $p_i^t$, and 0 otherwise. Then, given the vector of all agent positions $P_t$ at some time $t$, we define the coverage ratio as the ratio between areas of all points in the unoccupied domain $S \setminus O$ that are covered and the entire area,

$$C_t^{\text{Ratio}} = \frac{\int_{S \setminus O} \max_{c \in \{1, \ldots, n\}} \text{BSM}(p_i^t, c) \, dc}{\text{Area}(S \setminus O)}.$$

We calculate the moving distance of agent $i$ between its current and last locations, and the average moving distance at time step $t$, $D_t^{\text{Ave}}$ is averaging of all agents’ moving distances. The $D_t^{\text{Ave}}$ is defined formally:

$$D_t^{\text{Ave}} = \frac{\sum_{i=1}^n \|p_i^t - p_{i-1}^t\|}{n}.$$

More specifically, our goal now is to maximize $C_t^{\text{Ratio}}$ while keeping $D_t^{\text{Ave}}$ and the amount of communication required by the agents as small as possible. In particular, our work builds on the work of [21], where an algorithm is proposed to solve this problem but requires constant communication among the agents. Our main goal is to relax this assumption to improve efficiency of the network while still achieving good overall performance in terms of the metrics defined above.

III. EVENT-TRIGGERED ALGORITHM DESIGN

In this section, we design the event-triggered virtual force algorithm that allows agents to decide for themselves when communication with neighbors is necessary to complete the global task. The event-triggered virtual force algorithm has two components: a motion control law that determines the control input of each agent, and a decision control law that decides when communication is required.

A. Motion control law

We first design the motion control law that allows agents to determine how to move in the environment based on a virtual force approach. Each agent is exposed to three type of forces: 1. a neighbor force, $F_{ij}^t$, that could be an
attractive or a repulsive force, 2. an obstacle force, \( \vec{F}_{i,o}^{t} \), that is a repulsive force, 3. a boundary force, \( \vec{F}_{i,b}^{t} \), that is a repulsive force. These forces are heavily affected by the distance between an agent and neighbors, obstacles, and boundaries, respectively. We adopt the following force equations from [21], where the authors only consider deployment over a field of uniform density. Modifying the algorithm to deploy over non-uniform fields will be reserved for future work. The neighbor force \( \vec{F}_{i,j}^{t} \) is defined as

\[
\vec{F}_{i,j}^{t} = \begin{cases} 
0, & \text{if } d_{ij}^{t} > R_C, \\
K_A(d_{ij}^{t} - d_{ij}^{th})(\frac{p_i^{t} - p_j^{t}}{d_{ij}^{t}}), & \text{if } R_C \geq d_{ij}^{t} > d_{ij}^{th}, \\
K_B(d_{ij}^{th} - d_{ij}^{t})(\frac{p_i^{t} - p_j^{t}}{d_{ij}^{th}}), & \text{if } d_{ij}^{t} < d_{ij}^{th},
\end{cases}
\]

where \( d_{ij}^{th} = \sqrt{3R_C}/2 = \sqrt{3R_S} \), and \( K_A \) and \( K_B \) are constants. The \( d_{ij}^{t} \) is a distance between an agent and a neighbor. Agent \( j \) is a neighbor of agent \( i \) if and only if \( ||p_i^{t} - p_j^{t}|| \leq R_C \). If agent \( i \) has a neighbor within its communication range, the agent communicate with the neighbor to collect its current location. When the neighbor’s location is received, the agent \( i \) calculates the distance to the neighbor \( d_{ij}^{t} \).

In addition, the obstacle force \( \vec{F}_{i,o}^{t} \) is defined as

\[
\vec{F}_{i,o}^{t} = \begin{cases} 
0, & \text{if } d_{i,o}^{t} \geq d_{o}^{th}, \\
(K_{r1}(d_{o}^{th} - d_{i,o}^{t}), \alpha_{i,o} + \pi), & \text{if } d_{i,o}^{t} < d_{o}^{th},
\end{cases}
\]

where \( d_{o}^{th} = \sqrt{3R_S}/2 \) and \( \alpha_{i,o} \) is a constant. The \( d_{i,o}^{t} \) is the shortest distance between an agent and an obstacle that is within agent \( i \)’s sensing range. If agent \( i \) senses an obstacle, \( o_m \), the agent calculates the shortest distance to the obstacle \( d_{i,o}^{t} \).

Furthermore, The boundary force \( \vec{F}_{i,b}^{t} \) is defined as

\[
\vec{F}_{i,b}^{t} = \begin{cases} 
0, & \text{if } d_{i,b}^{t} \geq d_{b}^{th}, \\
(K_{r2}(d_{b}^{th} - d_{i,b}^{t}), \alpha_{i,b} + \pi), & \text{if } d_{i,b}^{t} < d_{b}^{th},
\end{cases}
\]

where \( d_{b}^{th} = \sqrt{3R_S}/2 \) and \( K_{r2} \) is a constant. The \( d_{i,b}^{t} \) is the perpendicular distance between an agent and a boundary that is within the agent’s sensing range. Let \( b \in \partial S \) be a point on the environment’s boundary, and the closed segment \([p_i^{t}, b] \subset S \) to be a line such that \([p_i^{t}, b] \perp \partial S \). If agent \( i \) senses an environment’s boundary, the agent calculates the perpendicular distance to the boundary. In a rectangle environment, the \( \vec{F}_{i,b}^{t} \) is the total sum of the four boundaries’ forces. Formally:

\[
\vec{F}_{i,b}^{t} = \vec{F}_{i,bx}^{t} + \vec{F}_{i,by}^{t} + \vec{F}_{i,bx}^{t} + \vec{F}_{i,by}^{t}
\]

Therefore, the Net-Force that attacks an agent at time \( t \), \( \vec{F}_{i}^{t} \), is the sum of all forces. The \( \vec{F}_{i}^{t} \) defined as:

\[
\vec{F}_{i}^{t} = \sum_{j \in N_i} \vec{F}_{i,j}^{t} + \sum_{m=1}^{N_0} \vec{F}_{i,o}^{t} + \vec{F}_{i,b}^{t},
\]

where \( N_i \subset P_0 \) is the set of agent \( i \)’s neighbors at time \( t \), and \( N_0 \) is the number of obstacles in the environment.

The Net-Force of an agent could be 0 if no neighbor is within the agent’s communication range and no obstacle and boundary are within its sensing range. Also, \( \vec{F}_{i}^{t} \) could be weak if the forces are contradictory or neighbors, obstacles and/or boundaries are far from an agent but within its communication and sensing ranges. On the other hand, \( \vec{F}_{i}^{t} \) could be strong if an agent has very close neighbors, obstacles and/or boundaries.

More specifically, \( \vec{F}_{i}^{t} \) is the desired displacement of agent \( i \) at position \( p_i^{t} \). Thus, it simply moves with velocity \( v_i^{t} \) towards this point by setting

\[
u_i^{t} = \frac{\vec{F}_{i}^{t}}{||\vec{F}_{i}^{t}||} v_i^{t},\]

where its velocity \( v_i^{t} \) is given by

\[
v_i^{t} = \min\{V_{max}, \alpha \frac{||\vec{F}_{i}^{t}||}{\Delta t}\},\]

where \( \alpha \in (0, 1) \).

The motion control law in short, at every instant of time, each agent calculates its \( \vec{F}_{i}^{t} \) and moves in the direction as fast as possible if \( ||\vec{F}_{i}^{t}|| > V_{max} \Delta t \). Otherwise, it moves in \( \vec{F}_{i}^{t} \) direction at slower speed. The simple motion control law is described formally in Algorithm 1

**Algorithm 1: motion control law**

Agent \( i \in \{1, \ldots, n\} \) performs at all times \( t \in \mathbb{Z}_{\geq 0} \):

1. receives positions \( p_j^{t} \) from neighbors \( j \) within a distance \( R_C \)
2. senses boundary \( \partial S \) and detects obstacles \( o_m \) within a distance \( R_S \)
3. computes \( \vec{F}_{i}^{t} \) according to (1)
4. sets \( v_i^{t} \) according to (3)
5. computes \( u_i^{t} \) according to (2)
6. computes \( p_{i+1}^{t} = p_i^{t} + u_i^{t} \)
7. moves to \( p_{i+1}^{t} \) by \( u_i^{t} \)

**B. Decision control law**

We are now interested in improving the motion control law which requires all agents to be in communication each time a control signal is computed by relaxing the need for constant communication. Let \( p_{event}^{t} \) be an intermediate goal that agent \( i \) can reach in multiple timesteps and define it as \( p_{event}^{t} = p_i^{t} + \vec{F}_{i}^{t} \). We aim to allow agents to travel towards \( p_{event}^{t} \) without constantly communicating with neighbors by designing the decision control law that combines two event-trigger conditions.

A trivial event-trigger condition, \textit{Condition1}, to be that an agent moves to \( p_{event}^{t} \) in multiple \( \Delta t \) without communicating with others. When the agent reaches \( p_{event}^{t} \), the agent communicates with neighbors to update its \( \vec{F}_{i}^{t} \), \( p_{event}^{t} \) and \( u_i^{t} \). However, this condition is problematic in some scenarios such as there could an obstacle blocking the way to \( p_{event}^{t} \) in \( 2\Delta t \).
To avoid collisions, we introduce an additional mechanism to trigger an event. Let \( X_i \in \mathbb{R}^2 \) be the union of neighbors, obstacles, and boundaries within agent \( i \)'s sensing range, and define \( R_T \) as the triggering radius. Then, given agent \( i \)'s current position \( p_i^t \) at time \( t \), we let a triggering sensing model (TSM) to be

\[
TSM(p_i^t, X_i) = \begin{cases} 
1 & \text{if } \exists x \in X_i \text{ s.t. } ||p_i^t - x|| \leq R_T \\
0 & \text{otherwise.}
\end{cases}
\]  

(4)

which returns 1 if agent \( i \) detects an object \( x \in X_i \) within this triggering range \( R_T \) of \( p_i^t \), and 0 otherwise. This triggering sensing model (TSM) does not guarantee no collisions unless \( R_T \) is bounded. In our analysis, the worst case scenario is that when two agents are traveling in the opposite direction by \( V_{\text{max}} \). The distance that will guarantee no collision between the agents must be more than \( 2V_{\text{max}}\Delta t \). Therefore, we lower bounded by \( R_T > 2V_{\text{max}}\Delta t \). For agents to move without a collision, \( \text{Condition 2} \), they are required to communicate with neighbors to update their \( F_i^t, P_i^t \) \text{event} \text{and} \( u_i \) if they sense an object within their triggering sensing range. Note, the smaller the \( R_T \), the less communication performed. Thus, we let \( R_T \) as small as possible. The \text{decision control law} combines both conditions and is described formally in Algorithm 2. 

\begin{algorithm}[h]
\caption{decision control law}
\begin{algorithmic}[1]
\STATE Agent \( i \in \{1, \ldots, n\} \) performs at every triggered event: 
\STATE 1: receives positions \( p_j^t \) from neighbors \( j \) within a distance \( R_C \)
\STATE 2: senses boundary \( \partial S \) and detects obstacles \( o_m \) within a distance \( R_S \)
\STATE 3: computes \( F_i^t \) according to (1)
\STATE 4: computes \( P_i^t \text{event} = p_i^t + F_i^t \)
\STATE 5: set \( TSM = 0 \)
\STATE 6: while \( (||p_i^t - p_j^t|| \neq 0 \& TSM \neq 1) \) do 
\STATE 7: computes \( p_i^{t+1} \)
\STATE 8: move to \( p_i^{t+1} \)
\STATE 9: set \( p_i^t = p_i^{t+1} \)
\STATE 10: sense all objects \( X_i \) within triggering sensing range
\STATE 11: compute TSM according to (4)
\STATE 12: end while
\end{algorithmic}
\end{algorithm}

C. The event-triggered virtual force algorithm

Here, we synthesize the event-triggered strategy that helps agents to determine at each \( \Delta t \) when updated information is needed to complete the task. Our designed algorithm is a combination of motion control law of Section III-A and \text{decision control law} of Section III-B with a procedure to acquire communication when the conditions are met.

[Informal description]: Agent \( i \) communicates with neighbors to collect their locations, and senses the surrounding to locate obstacles and boundaries that are within its sensing range. When, the attractive and/or repulsive forces are calculated, the agent computes its \( F_i^t \). When \( F_i^t \) is computed, the agent calculates \( p_i^t \text{event} \). It sets its \( v_i = \min(V_{\text{max}}, \|F_i^t\|/\Delta t) \), and computes its \( u_i \). Then, the agent calculates \( p_i^{t+1} \) \text{and} moves towards it. When \( p_i^{t+1} \) reached, the agent senses for an object within its triggering sensing range and updates \( TSM \). If the TSM returned 0, the agent calculates a new \( p_i^{t+1} \), moves to \( p_i^{t+1} \), senses for an object at \( p_i^{t+1} \) until it reaches the \( p_i^{t+1} \text{event} \). If \( p_i^{t+1} \text{event} \) is reached or the TSM returned 1, the agent communicates with neighbors to update its \( F_i^t, p_i^{t+1} \text{event} \text{and} u_i \).

The event-triggered virtual force algorithm is described formally in Algorithm 3.

\begin{algorithm}[h]
\caption{event-triggered virtual force algorithm}
\begin{algorithmic}[1]
\STATE Agent \( i \in \{1, \ldots, n\} \) performs at every triggered event: 
\STATE 1: receives positions \( p_j^t \) from neighbors \( j \) within a distance \( R_C \)
\STATE 2: senses boundary \( \partial S \) and detects obstacles \( o_m \) within a distance \( R_S \)
\STATE 3: computes \( F_i^t \) according to (1)
\STATE 4: computes \( P_i^t \text{event} = p_i^t + F_i^t \)
\STATE 5: set \( TSM = 0 \)
\STATE 6: while \( (||p_i^t - p_j^t|| \neq 0 \& TSM \neq 1) \) do 
\STATE 7: if \( ||u_i|| \leq ||p_i^t - p_j^t|| \) then
\STATE 8: computes \( p_i^{t+1} = p_i^t + u_i \)
\STATE 9: moves to \( p_i^{t+1} \) \text{by} \( v_i \)
\STATE 10: updates \( p_i^t = p_i^{t+1} \)
\STATE 11: sense all objects \( X_i \) within triggering sensing range
\STATE 12: compute TSM according to (4)
\STATE 13: end if
\STATE 14: end while
\end{algorithmic}
\end{algorithm}

IV. SIMULATION

In this section, we provide simulations of the event-triggered virtual force algorithm. All simulations are done in MATLAB with \( n = 40 \) agents moving in \( 100m \) by \( 100m \) square environment and \( 300 \) time steps. All simulation are averaged over \( 100 \) runs, and only few simulations are shown due to limited paper number. The parameters of the simulations are given in TABLE 1.

\begin{table}[h]
\caption{Simulation parameter}
\begin{tabular}{|l|l|l|}
\hline
\text{grid size} & \text{100mX100m} & \text{100mX100m} \\
\hline
\text{K_C} & \text{0.8} & \text{0.8} \\
\text{K_A} & \text{0.1} & \text{0.2} \\
\text{R_S} & \text{10m} & \text{20m} \\
\text{R_T} & \text{1.1m} & \text{3.2m} \\
\hline
\end{tabular}
\end{table}

We adopt a communication power model from [30]. Specifically, the total power \( P_i \) used by agent \( i \) to communicate, in \( dBm/W \) power units is defined as:

\[
P_i = 10 \log_{10} \left[ \sum_{j=1}^{n} \beta_1 10^{0.1P_{i-j} + \beta_2 ||p_i - p_j'||} \right]
\]

where \( \beta_1 \) and \( \beta_2 \) are positive real parameters that depend on the characteristics of the wireless medium, and \( P_{i-j} \) is the power received by agent \( j \) of the signal transmitted by agent \( i \). In our simulations, all these values are set to 1.

We start by comparing the performance of event-triggered virtual force algorithm with OAVFA in [21]. The event-triggered virtual
force algorithm achieves the final deployment as in [21]. Figure 1 illustrate the initial and final deployment of the event-triggered virtual force algorithm in the presence of an obstacle. Our algorithm achieves maximum coverage area of the environment. This initialization of the agents is based on randomly placing ten agents in each corner of the environment, and note that all further illustrated results assume this initialization.

![Fig. 1](image)

Fig. 1. (a) Initial deployment of 40 agents in the environment with an obstacle. (b) Final deployment of (a) initialization using event-triggered virtual force algorithm.

Figures 2 (a) and (b) compare the $C_1^{\text{Ratio}}$ performance of event-triggered virtual force algorithm and OAVFA without and with an obstacle. The result shows that both algorithms achieve maximum area coverage, but the OAVFA reaches the maximum slightly faster. Figures 2 (c) and (d) compares the $D_1^{\text{Ave}}$ performance without and with an obstacle. The results show that our algorithm has a better performance. In case of no obstacles, our algorithm has slightly better $D_1^{\text{Ave}}$, but in case of the presence of an obstacle, our performance minimized $D_1^{\text{Ave}}$ more than OAFVA.

Figures 2 (e) and (f) illustrate the average communication power consumed without and with an obstacle. The figures show the contribution of our work. The event-triggered virtual force algorithm saved unnecessary communication between agents that resulted in significant reduction of the communication power consumed. The average power saved without obstacles is more than 50%, and the average power saved in the presence of an obstacle is more than 54%.

V. CONCLUSIONS

In this paper we considered a multi-agent coverage control problem in the presence of obstacles. To solve the problem, we have proposed the event-triggered virtual force algorithm by combining the motion control law and the decision control law which allows agents to autonomously decide for themselves when communication is required, in addition to how to move. Our algorithm allows agents to travel in the environment without the need to communicate with neighbors at every instant of time. An agent only needs to communicate if it reaches its new location or if it senses an object within the triggering sensing range. The main contribution of this work is reducing the amount of communication between agents while maintaining the desired coverage control performance. Our simulations illustrated that the event-triggered virtual force algorithm had reduced more than 50% of communication power compared to the OAVFA in [21] with and without the presence of obstacles, and also achieved maximum area coverage as if agents communicated periodically. Future work will be devoted to include scenarios such as communication delays and package drops with grantees on the level of performance and power saving. In addition, we are interested in developing asynchronous implementation by identifying event trigger condition that ensure the level of performance.

REFERENCES


Fig. 2. A comparison of the CRatio between event-triggered virtual force algorithm and OAVFA in [21] (a) without obstacles and (b) with an obstacle. A comparison of the DAv between our algorithm and OAVFA in [21] (c) without obstacles and (d) with an obstacle. A comparison of the the average communication power consumed between our algorithm and OAVFA (e) without obstacles and (f) with an obstacle


