Self-triggered coordination of robotic networks for optimal deployment

Cameron Nowzari   Jorge Cortés

***Abstract*— This paper studies a coverage control problem for multi-vehicle systems where individual agents operate with outdated information about each others' locations. Our objective is to understand to what extent this outdated information is still useful and at which point it becomes essential to obtain new, up-to-date information. We propose a self-triggered coordination algorithm based on spatial partitioning techniques with uncertain information and verify its correctness using tools from computational geometry, stability theory, set-valued analysis, and event-based systems.**

I. Introduction

This paper studies a robotic sensor network performing an optimal static deployment task when individual agents do not have up-to-date information about each others' locations. Our objective is to design a self-triggered coordination algorithm that allows agents to decide autonomously when new, up-to-date location information is needed to complete the task. Our motivation comes from the need for strategies that naturally account for uncertainty in the state of other agents caused by, for instance, sparse communication and sensor errors.

Literature review: In the context of robotic sensor networks, this work builds on [1], where distributed algorithms based on centroidal Voronoi partitions are presented, see also [2]. Voronoi partitions are also employed in [3], [4], [5]. Other works on deployment coverage problems include [4], [6].

A feature of the algorithms mentioned above is the common assumption of constant communication among agents and fresh, up-to-date information about each others' locations. The other areas of relevance to this work are discrete-event systems [7], self-triggered control [8], [9], [10], [11] and event-triggered control [12], [13], [14], [15] of sensor and actuator networks. These works trade computation at the agent level for less communication, sensing or actuator effort while still guaranteeing a desired level of performance.

Statement of contributions: The main contribution of the paper is the design of the `self-triggered centroid algorithm` to achieve optimal static deployment in a given convex environment. We first design an update policy that helps an agent determine under what conditions the location information it possesses about other agents is sufficiently up-to-date. This policy is based on spatial partitioning techniques with uncertain information, and in particular, on the notions of guaranteed Voronoi and a new notion we call the dual guaranteed Voronoi diagram. We then design a motion control law that, given the (possibly outdated) information an agent has, determines a motion plan that is guaranteed to contribute positively to achieving the deployment task. We establish the monotonic evolution of the aggregate objective function encoding the notion of deployment and characterize the convergence properties of the algorithm. Due to the discontinuous nature of the data structure that agents maintain in our self-triggered coordination law, the technical approach resorts to a combination of notions and tools from computational geometry, set-valued analysis, and stability theory. Various simulations illustrate the performance and implementation cost of the `self-triggered centroid algorithm`. For reasons of space, all proofs are omitted. The interested reader is referred to [16].

The authors are with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA 92093, USA, {cnowzari,cortes}@ucsd.edu

II. Preliminaries

We let $\mathbb{R}_{\geq 0}$ and $\mathbb{Z}_{\geq 0}$ be the sets of nonnegative real and integer numbers, respectively, and $\|\cdot\|$ be the Euclidean distance.

A. Basic geometric notions

We denote by $[p, q] \subset \mathbb{R}^d$ the closed segment with extreme points p and $q \in \mathbb{R}^d$. Let $\phi : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ be a bounded measurable function that we term *density*. For $S \subset \mathbb{R}^d$, the *mass* and *center of mass* of S with respect to ϕ are

$$M_S = \int_S \phi(q)dq, \qquad C_S = \frac{1}{M_S}\int_S q\phi(q)dq.$$

Given $v \in \mathbb{R}^d \setminus \{0\}$, let $\mathrm{unit}(v)$ be the unit vector in the direction of v. Given a convex set $S \subset \mathbb{R}^d$ and $p \in \mathbb{R}^d$, let $\mathrm{pr}_S(p)$ denote the orthogonal projection of p onto S, i.e., $\mathrm{pr}_S(p)$ is the point in S closest to p. The *to-ball-boundary* map $\mathrm{tbb} : (\mathbb{R}^d \times \mathbb{R}_{\geq 0})^2 \to \mathbb{R}^d$ takes (p, δ, q, r) to

$$\begin{cases} p + \delta\,\mathrm{unit}(q-p) & \text{if } \|p - \mathrm{pr}_{\overline{B}(q,r)}(p)\| \geq \delta, \\ \mathrm{pr}_{\overline{B}(q,r)} & \text{if } \|p - \mathrm{pr}_{\overline{B}(q,r)}(p)\| \leq \delta. \end{cases}$$

Figure 1 illustrates the action of tbb.

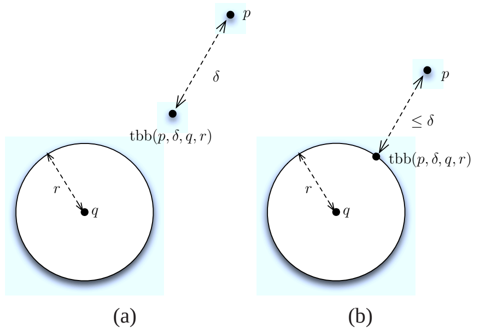


Fig. 1. Graphical representation of the action of tbb when (a) $\|p - \mathrm{pr}_{\overline{B}(q,r)}(p)\| > \delta$ and (b) $\|p - \mathrm{pr}_{\overline{B}(q,r)}(p)\| \leq \delta$.

The *circumcenter* of $S \subset \mathbb{R}^d$, denoted $\mathrm{cc}(S)$, is the center of the closed ball of minimum radius that contains S. The *circumradius* of S, denoted $\operatorname{cr}(S)$, is the radius of this ball. We denote by $\overline{B}(p,r)$ the closed ball centered at $p \in S$ with radius r and by $H_{po} = \{q \in \mathbb{R}^d \mid \|q-p\| \le \|q-o\|\}$ the closed halfspace determined by $p, o \in \mathbb{R}^d$ that contains p.

B. Voronoi partitions

We refer to [17] for a comprehensive treatment on Voronoi partitions and briefly present some relevant concepts here. Let S be a convex polygon in $\mathbb{R}^2$ and $P = (p_1, \dots, p_n)$ be the location of n sensors. A *partition* of S is a collection of n polygons $\mathcal{W} = \{W_1, \dots, W_n\}$ with disjoint interiors whose union is S. The *Voronoi partition* $\mathcal{V}(P) = \{V_1, \dots, V_n\}$ of S generated by the points $P = (p_1, \dots, p_n)$ is

$$V_i = \{q \in S \mid \|q - p_i\| \le \|q - p_j\|,\ \forall j \neq i\}.$$

When the Voronoi regions V_i and V_j are adjacent (i.e., they share an edge), p_i is called a *(Voronoi) neighbor* of p_j (and vice versa). We denote the neighbors of agent i by $\mathcal{N}_i$. $P = (p_1, \dots, p_n)$ is a *centroidal Voronoi configuration* if it satisfies that $p_i = C_{V_i}$, for all $i \in \{1, \dots, n\}$.

C. Facility location and aggregate distortion

We briefly introduce a locational optimization function called aggregate distortion, see [18], [2], that is key in the design and analysis of our algorithm. Consider a set of sensors with positions P in an environment S. The *sensing performance* at point q taken from the ith sensor at p_i degrades with the squared distance $\|q - p_i\|^2$. Assume also that a density function $\phi : S \to \mathbb{R}$ is available, so that $\phi(q)$ reflects the possibility of an event happening at position q. Consider the task of minimizing the locational optimization function

$$\mathcal{H}(P) = E_\phi \left[\min_{i \in \{1,\dots,n\}} \|q - p_i\|^2 \right]. \tag{1}$$

It is interesting to note that this function can be rewritten in terms of the Voronoi partition as

$$\mathcal{H}(P) = \sum_{i=1}^{n} \int_{V_i} \|q - p_i\|^2 \phi(q) dq,$$

This suggests defining a generalization of $\mathcal{H}$, which with a slight abuse of notation we denote by the same letter, as

$$\mathcal{H}(P, \mathcal{W}) = \sum_{i=1}^{n} \int_{W_i} \|q - p_i\|^2 \phi(q) dq, \tag{2}$$

where $\mathcal{W}$ is a partition of S, and the ith sensor is responsible of the "dominance region" W_i. Note that $\mathcal{H}(P) = \mathcal{H}(P, \mathcal{V}(P))$. The function $\mathcal{H}$ is to be minimized with respect to both the sensors' locations P and the assignment of the dominance regions $\mathcal{W}$. The following result [18], [2] characterizes its critical points.

Lemma II.1 *Given $P \in S^n$ and a partition $\mathcal{W}$ of S,*

$$\mathcal{H}(P, \mathcal{V}(P)) \le \mathcal{H}(P, \mathcal{W}), \tag{3}$$

i.e., the optimal partition is the Voronoi partition. Moreover, for $P' \in S^n$ with $\|p'_i - C_{W_i}\| \le \|p_i - C_{W_i}\|$, $i \in \{1, \dots, n\}$,

$$\mathcal{H}(P', \mathcal{W}) \le \mathcal{H}(P, \mathcal{W}),$$

i.e., the optimal sensor positions are the centroids.

III. Problem statement

Consider a group of agents moving in a convex polygon $S \subset \mathbb{R}^2$ with positions $p_1, \dots, p_n$. For simplicity, we consider first-order continuous-time dynamics, although our treatment could be extended to arbitrary controllable dynamics with minimal modifications. Specifically,

(i) all agents' clocks are synchronous, i.e., given a common starting time t_0, subsequent timesteps occur for all agents at $t_\ell = t_0 + \ell \Delta t$, for $\ell \in \mathbb{Z}_{\ge 0}$, and
(ii) each agent can move a maximum amount of $v_{\max}$ in one second, i.e., $\|p_i(t_{\ell+1}) - p_i(t_\ell)\| \le v_{\max} \Delta t$.

For simplicity of presentation, we consider the case of a common maximum velocity bound $v_{\max}$ for all agents. The results of the paper are extensible to the case when each agent has its own maximum velocity bound.

Our objective is to achieve optimal deployment, measured according to the expected-value measure $\mathcal{H}$ introduced in (1), even when agents have uncertain information about each others' positions. Because the cost to communicate increases with distance, agents might need to balance the need for up-to-date location information with the need to spend as little energy as possible. Our goal is to understand the trade-offs between deployment performance and communication cost.

The data structure that each agent i maintains about other agents j is the last known location p^i_j and the time elapsed $\tau^i_j \in \mathbb{R}_{\ge 0}$ since this information was received, for each $j \in \{1, \dots, n\} \setminus \{i\}$. For itself, agent i has access to up-to-date location information, i.e., $p^i_i = p_i$ and $\tau^i_i = 0$ at all times. With this data, agent i knows that, at the current time, agent j will not have traveled more than a distance $r^i_j = v_{\max} \tau^i_j$ from p^i_j, and hence agent i can construct a ball $\overline{B}(p^i_j, r^i_j)$ that is guaranteed to contain the actual location of agent j. This data is stored in the vector

$$\mathcal{D}^i = ((p^i_1, r^i_1), \dots, (p^i_n, r^i_n)) \in (S \times \mathbb{R}_{\ge 0})^n.$$

Additionally, agent i maintains a variable $\mathcal{A}^i \subset \{1, \dots, n\}$ with $i \in \mathcal{A}^i$ that, at any time t, corresponds to the agents whose position information should be used. For instance, $\mathcal{A}^i = \{1, \dots, n\}$ would mean that agent i uses all the information contained in $\mathcal{D}^i$. As we will explain in Section V-B, this is not always necessary. We refer to $\mathcal{D} = (\mathcal{D}^1, \dots, \mathcal{D}^n) \in (S \times \mathbb{R}_{\ge 0})^{n^2}$ as the entire memory of the network. We find it convenient to define the map $\operatorname{loc} : (S \times \mathbb{R}_{\ge 0})^{n^2} \to S^n$ to extract the exact agents' location information from $\mathcal{D}$ by $\operatorname{loc}(\mathcal{D}) = (p^1_1, \dots, p^n_n)$.

To optimize $\mathcal{H}$, the knowledge of its own Voronoi cell is critical to each agent, cf. Section II-C. However, with the data structure described above, agents cannot compute the Voronoi partition exactly. Instead, they implement the space partitioning techniques described in the following section.

IV. Space partitioning with uncertainty

Because we are interested in applications in which the available information is not perfect, we introduce here spatial partitioning techniques with uncertain information. Here, we review the concept of a guaranteed Voronoi diagram as in [19] and present a new tool we call the dual guaranteed Voronoi diagram. Let $S \subset \mathbb{R}^2$ be a domain and consider a set of regions $D_1, \dots, D_n \subset S$, each containing a site $p_i \in D_i$. The *guaranteed Voronoi diagram* of S generated by $D_1, \dots, D_n$ is the collection of sets $\mathrm{g}\mathcal{V}(D_1, \dots, D_n) = \{\mathrm{g}V_1, \dots, \mathrm{g}V_n\}$ defined by

$$\mathrm{g}V_i = \{q \in S \mid \max_{x \in D_i} \|q - x\| \le \min_{y \in D_j} \|q - y\| \text{ for all } j \neq i\}.$$

With a slight abuse of notation, we denote by $\mathrm{g}V_i(D)$ the ith component of $\mathrm{g}\mathcal{V}(D_1, \dots, D_n)$. The interpretation of $\mathrm{g}\mathcal{V}(D_1, \dots, D_n)$ is the following: $\mathrm{g}V_i$ contains the points of S that are guaranteed to be closer to p_i than to any other of the nodes p_j, $j \neq i$. Because the information about the location of these nodes is uncertain, there is a neutral region in S which is not assigned to anybody: those points for which no guarantee can be established. Unlike the standard Voronoi partition, the guaranteed Voronoi diagram is not a partition of S, see Figure 2(a). Each point in the boundary of $\mathrm{g}V_i$

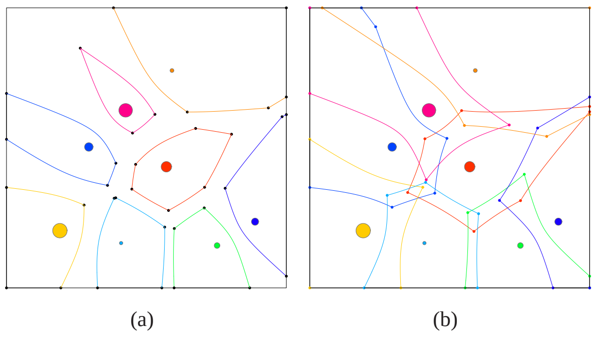


Fig. 2. Guaranteed Voronoi (a) and dual guaranteed Voronoi (b) diagrams.

belongs to a set of the form

$$\Delta^{\mathrm{g}}_{ij} = \{q \in S \mid \max_{x \in D_i} \|q - x\| = \min_{y \in D_j} \|q - y\|\}, \qquad (4)$$

for some $j \neq i$. Note that in general $\Delta^{\mathrm{g}}_{ij} \neq \Delta^{\mathrm{g}}_{ji}$.
On the other hand, the *dual guaranteed Voronoi diagram* of S generated by $D_1, \dots, D_n$ is the collection of sets $\mathrm{dg}\mathcal{V}(D_1, \dots, D_n) = \{\mathrm{dg}V_1, \dots, \mathrm{dg}V_n\}$ defined by

$$\mathrm{dg}V_i = \{q \in S \mid \min_{x \in D_i} \|q - x\| \le \max_{y \in D_j} \|q - y\| \text{ for all } j \neq i\}.$$

We denote by $\mathrm{dg}V_i(D)$ the ith component of $\mathrm{dg}\mathcal{V}(D_1, \dots, D_n$. The interpretation of $\mathrm{dg}\mathcal{V}(D_1, \dots, D_n)$ is the following: the points of S outside $\mathrm{dg}V_i$ are guaranteed to be closer to some other node p_j, $j \neq i$ than to p_i. Because the information about the location of these nodes is uncertain, there are regions of the space that belong to more than one cell. The dual guaranteed Voronoi diagram is a covering of the set S, see Figure 2(b). Each point in the boundary of $\mathrm{dg}V_i$ belongs to a set of the form

$$\Delta^{\mathrm{dg}}_{ij} = \{q \in S \mid \min_{x \in D_i} \|q - x\| = \max_{y \in D_j} \|q - y\|\}, \qquad (5)$$

for some $j \neq i$. Note that in general $\Delta^{\mathrm{dg}}_{ij} \neq \Delta^{\mathrm{dg}}_{ji}$.

If every region D_i is a point, $D_i = \{p_i\}$, then $\mathrm{g}V_i$ and $\mathrm{dg}V_i$ coincide with the standard Voronoi cell V_i of p_i, and the guaranteed and dual guaranteed Voronoi diagrams are the Voronoi partition of S generated by $p_1, \dots, p_n$. In general, for any collection of points $p_i \in D_i$, $i \in \{1, \dots, n\}$, it holds that $\mathrm{g}V_i \subset V_i \subset \mathrm{dg}V_i$, $i \in \{1, \dots, n\}$. Agent p_j is a guaranteed Voronoi neighbor of p_i if $\Delta^{\mathrm{g}}_{ij} \cap \partial \mathrm{g}V_i \neq \emptyset$. The set of guaranteed Voronoi neighbors of agent i is denoted by $\mathrm{g}\mathcal{N}_i(D)$, where we use the notation $D = (D_1, \dots, D_n)$.
Throughout the paper, we consider uncertain regions given by balls, $D_i = \overline{B}(p_i, r_i)$, $i \in \{1, \dots, n\}$. In this case, the edges composing the boundary of $\mathrm{g}V_i$ in (4) are of the form,

$$\Delta^{\mathrm{g}}_{ij} = \{q \in S \mid \|q - p_i\| + r_i = \|q - p_j\| - r_j\}, \qquad (6)$$

and therefore, lie on the arm of the hyperbola closest to p_i with foci p_i and p_j, and semimajor axis $\frac{1}{2}(r_i + r_j)$. The edges composing the boundary of $\mathrm{dg}V_i$ in (5) are of the form,

$$\Delta^{\mathrm{dg}}_{ij} = \{q \in S \mid \|q - p_i\| - r_i = \|q - p_j\| + r_j\}, \qquad (7)$$

and therefore, lie on the arm of the hyperbola farthest from p_i with foci p_i and p_j, and semimajor axis $\frac{1}{2}(r_i + r_j)$. The following results state useful properties of the guaranteed and dual guaranteed Voronoi diagrams.

Lemma IV.1 *Given $p_1, \dots, p_n \in S$ and $r_1, \dots, r_n, a \in \mathbb{R}_{\ge 0}$, let $D_i = \overline{B}(p_i, r_i)$ and $D'_i = \overline{B}(p_i, r_i + a)$, for $i \in \{1, \dots, n\}$. Then, $\mathrm{g}\mathcal{N}_i(D'_1, \dots, D'_n) \subset \mathrm{g}\mathcal{N}_i(D_1, \dots, D_n)$, for all $i \in \{1, \dots, n\}$.*

Lemma IV.2 *Given sets $D_1, \dots, D_{n+m} \subset S$, it holds that $\mathrm{dg}V_i(D_1, \dots, D_n, D_{n+1}, \dots, D_{n+m}) \subseteq \mathrm{dg}V_i(D_1, \dots, D_n)$ for all $i \in \{1, \dots, n\}$.*

V. Self-triggered coverage optimization

Here we design a coordination strategy to solve the problem described in Section III. From the point of view of an agent, the algorithm is composed of two parts: a motion control component that determines the best way to move given the available information and an update decision component that determines when new information should be obtained.

A. Motion control

If an agent had perfect knowledge of other agents' positions, then to optimize $\mathcal{H}$, it could compute its own Voronoi cell and move towards its centroid, as in [1]. Since this is not the case, we instead propose an alternative motion control law. Let us describe it first informally:

> *[Informal description]:* At each round, each agent uses its stored information about other agents' locations to calculate its own guaranteed and dual guaranteed Voronoi cells. Then, the agent moves towards the centroid of its guaranteed Voronoi cell.

In general, there is no guarantee that following the `motion control law` will lead the agent to get closer to the centroid of its Voronoi cell. A condition under which this statement holds is characterized by the following result.

Lemma V.1 *Given $p \neq q, q^* \in \mathbb{R}^2$, let $p' \in [p, q]$ such that $\|p' - q\| \geq \|q^* - q\|$. Then, $\|p' - q^*\| \leq \|p - q^*\|$.*

Therefore, with the notation of Lemma V.1, if agent i is at $p = p_i$, computes the target $q = C_{\mathrm{g}V_i}$ and moves towards it to p', then the distance to $q^* = C_{V_i}$ decreases as long as

$$\|p' - C_{\mathrm{g}V_i}\| \geq \|C_{V_i} - C_{\mathrm{g}V_i}\| \tag{8}$$

holds. The right-hand side cannot be computed exactly by i because of lack of information about C_{V_i}. However, the distance between the centroids of the guaranteed Voronoi and Voronoi cells can be upper bounded, as we show next.

Proposition V.2 *Let $L \subset V \subset U$. Then, for any density function ϕ, the following holds*

$$\|C_V - C_L\| \leq 2\,\mathrm{cr}(U)\Big(1 - \frac{M_L}{M_U}\Big).$$

With the notation of Proposition V.2, agent i can use $L = \mathrm{g}V_i$ and $U = \mathrm{dg}V_i$ to upper bound the distance $\|C_{V_i} - C_{\mathrm{g}V_i}\|$ by

$$\mathrm{bnd}_i \equiv \mathrm{bnd}(\mathrm{g}V_i, \mathrm{dg}V_i) = 2\,\mathrm{cr}(\mathrm{dg}V_i)\Big(1 - \frac{M_{\mathrm{g}V_i}}{M_{\mathrm{dg}V_i}}\Big). \tag{9}$$

This bound is computable with the information stored in its own memory $\mathcal{D}^i$. Agent i can use this bound to guarantee that the condition (8) holds by making sure that

$$\|p' - C_{\mathrm{g}V_i}\| \geq \mathrm{bnd}_i \tag{10}$$

holds. The point p' to which agent i moves to is determined as follows: move towards $C_{\mathrm{g}V_i}$ as much as possible in one time step until it is within distance bnd_i of it. Formally, the `motion control law` is described in Algorithm 1.

Algorithm 1: `motion control law`

```
Agent $i \in \{1, \dots, n\}$ performs:
 1: set $D = \mathcal{D}^i$
 2: compute $L = \mathrm{g}V_i(D)$ and $U = \mathrm{dg}V_i(D)$
 3: compute $q = C_L$ and $r = \mathrm{bnd}(L, U)$
 4: move to $\mathrm{tbb}(p_i, v_{\max}\Delta t, q, r)$
 5: set $\mathcal{D}^i_j = (p^i_j, r^i_j + v_{\max}\Delta t)$
 6: set $\mathcal{D}^i_i = (\mathrm{tbb}(p_i, v_{\max}, q, r), 0)$
```

Clearly, if time elapses without new location information, then the bound (9) grows larger and (10) becomes harder to satisfy until it becomes unfeasible. Therefore, agents need an decision mechanism that establishes when new information is required for the execution of the motion control law to achieve its objective. This is addressed in Section V-B.

B. Update decision policy

The second component of the self-triggered strategy takes care of updating the memory of the agents, and in particular, of deciding when new information is needed. This is essentially achieved by making sure that (10) is feasible. Two reasons can make (10) invalid for a given agent i. On the one hand, the bound bnd_i might be large due to outdated location information about other agents' location in $\mathcal{D}^i$. This should trigger the need for up-to-date information through communication with other agents. On the other hand, agent i might be close to $C_{\mathrm{g}V_i}$, requiring bnd_i to be small in order for the condition to hold. We deal with this by specifying a tolerance $\varepsilon > 0$ that can be selected a priori by the designer. Formally, the memory updating mechanism followed by each agent is described by the pseudo-code in Algorithm 2.

Algorithm 2: `one-step-ahead update policy`

```
Agent $i \in \{1, \dots, n\}$ performs:
 1: set $D = \mathcal{D}^i$
 2: compute $L = \mathrm{g}V_i(D)$ and $U = \mathrm{dg}V_i(D)$
 3: compute $q = C_L$ and $r = \mathrm{bnd}(L, U)$
 4: if $r \geq \max\{\|q - p_i\|, \varepsilon\}$ then
 5:     reset $\mathcal{D}^i$ by acquiring up-to-date location information
 6: end if
```

According to Algorithm 2, agent i checks at each time step if condition (10) is feasible or $\mathrm{bnd}_i \leq \varepsilon$, and therefore it is advantageous to execute the `motion control law` for one timestep. One could also implement a refined version of this decision policy making use of the fact that agent i has all the information it requires to perform this check for multiple steps into the future.

C. The `self-triggered centroid algorithm`

The self-triggered coordination algorithm is the result of combining the motion control law of Section V-A and the update policy of Section V-B with a procedure to acquire up-to-date information about other agents when this requirement is triggered (cf. `5:` in Algorithm 2). A trivial update mechanism will be to provide each agent with up-to-date information about the location of all other agents in the network; however, this is costly from a communications point of view. Instead, we propose an alternative algorithm that only provides up-to-date location information of the Voronoi neighbors at the specific time when step `5:` is executed. The `Voronoi cell computation` is borrowed from [1]. We present it in Algorithm 3, adapted to our scenario.

Algorithm 3: `Voronoi cell computation`

```
 1: initialize $R_i = \min_{k \in \{1,\dots,n\}\setminus\{i\}} \|p_i - p^i_k\| + v_{\max}\tau^i_k$
 2: detect all $p_j$ within radius $R_i$
 3: set $W(p_i, R_i) = \overline{B}(p_i, R_i) \cap \big(\cap_{j:\|p_i - p_j\| \leq R_i} H_{p_i p_j}\big)$
 4: while $R_i < 2\max_{q \in W(p_i, R_i)} \|p_i - q\|$ do
 5:     set $R_i := 2R_i$
 6:     detect all $p_j$ within radius $R_i$
 7:     set $W(p_i, R_i) = \overline{B}(p_i, R_i) \cap \big(\cap_{j:\|p_i - p_j\| \leq R_i} H_{p_i p_j}\big)$
 8: end while
 9: set $V_i = W(p_i, R_i)$
10: set $\mathcal{A}^i = \mathcal{N}_i \cup \{i\}$ and $\mathcal{D}^i_j = (p_j, 0)$ for $j \in \mathcal{N}_i$
```

The `Voronoi cell computation` is based on the agent gradually increasing its communication radius until all the information required to construct its exact Voronoi cell has been obtained. It can be shown [16] that an agent i can compute the sets L and U in the algorithms described above with the information provided by Algorithm 3.

The combination of Algorithms 1-3 leads to the synthesis of the `self-triggered centroid algorithm` described in Algorithm 4 ($\pi_{\mathcal{A}^i}$ denotes the map that extracts from $\mathcal{D}^i$ the information about the agents contained in $\mathcal{A}^i$).

VI. Convergence analysis

In this section, we analyze the asymptotic convergence properties of `self-triggered centroid algorithm`.

Algorithm 4: `self-triggered centroid algorithm`

```
Initialization
 1: set $\mathcal{D}^i$ and $\mathcal{A}^i$ by running Voronoi cell computation
 2: set $\mathcal{D}^i_i = (p_i, 0)$
At timestep $\ell$, agent $i \in \{1,\dots,n\}$ performs:
 1: set $D = \pi_{\mathcal{A}^i}(\mathcal{D}^i)$
 2: compute $L = \mathrm{g}V_i(D)$ and $U = \mathrm{dg}V_i(D)$
 3: compute $q = C_L$ and $r = \mathrm{bnd}(L, U)$
 4: if $r \geq \max\{\|q - p_i\|, \varepsilon\}$ then
 5:     reset $\mathcal{D}^i$ and $\mathcal{A}^i$ by running Voronoi cell computation
 6:     set $D = \pi_{\mathcal{A}^i}(\mathcal{D}^i)$
 7:     set $L = \mathrm{g}V(D)$ and $U = \mathrm{dg}V(D)$
 8:     set $q = C_L$ and $r = \mathrm{bnd}(L, U)$
 9: end if
10: move to $\mathrm{tbb}(p_i, v_{\max}, q, r)$
11: set $\mathcal{D}^i_i = (\mathrm{tbb}(p_i, v_{\max}, q, r), 0)$
12: set $\mathcal{D}^i_j = (p^i_j, r^i_j + v_{\max}\Delta t)$ for $j \neq i$
```

An extension of the algorithm that makes agents decrease their maximum speed as the network gets closer to the set of centroidal Voronoi configurations is explored in [16]. The case of a constant maximum velocity is analyzed here. Note that this algorithm can be written as a map $f_{\mathrm{stca}} : (S \times \mathbb{R}_{\geq 0})^{n^2} \to (S \times \mathbb{R}_{\geq 0})^{n^2}$ which corresponds to the composition of a "decide/acquire-up-to-date-information" map f_{info} and a "move-and-update-uncertainty" map f_{motion}, i.e., $f_{\mathrm{stca}}(\mathcal{D}) = f_{\mathrm{motion}}(f_{\mathrm{info}}(\mathcal{D}))$ for $\mathcal{D} \in (S \times \mathbb{R}_{\geq 0})^{n^2}$. Our analysis strategy here is shaped by the fact that f_{info}, and consequently, f_{stca} are discontinuous. Our objective is to prove the following result characterizing the asymptotic convergence properties of the trajectories of the algorithm.

Proposition VI.1 *For $\varepsilon \in [0, \mathrm{diam}(S))$, the agents' position evolving under the `self-triggered centroid algorithm` from any initial network configuration in S^n converges to the set of centroidal Voronoi configurations.*

Given that the map f_{stca} is discontinuous, we cannot apply the discrete-time LaSalle Invariance Principle. Our proof strategy consists of constructing a closed set-valued map T, whose trajectories include the ones of f_{stca}, and apply the LaSalle Invariance Principle for set-valued maps [2]. For reasons of space, we do not include the full proof and only provide a sketch. The interested reader is referred to [16].

The definition of T is as follows. For convenience, we recall that $\mathcal{D} = (\mathcal{D}^1, \dots, \mathcal{D}^n) \in (S \times \mathbb{R}_{\geq 0})^{n^2}$, and that the elements of $\mathcal{D}^i$ are referred to as $((p^i_1, r^i_1), \dots, (p^i_n, r^i_n))$. To ease the exposition, we divide the construction of T in two steps, a first one that captures the agent motion and the uncertainty update to the network memory, and a second one that captures the acquisition of up-to-date network information.

Motion and uncertainty update: Note that once the uncertainty radius about the position of an agent hits the diameter of S, it does not need to grow anymore. This justifies the definition of the continuous motion map $\mathcal{M} : (S \times \mathbb{R}_{\geq 0})^{n^2} \to (S \times \mathbb{R}_{\geq 0})^{n^2}$ whose ith component is

$$\begin{aligned}\mathcal{M}_i(\mathcal{D}) = \big(&(p^i_1, \min\{r^i_1 + v_{\max}\Delta t, \mathrm{diam}(S)\}), \dots, \\ &(\mathrm{tbb}(p^i_i, v_{\max}, C_{\mathrm{g}V_i}(\pi_{\mathcal{A}^i}(\mathcal{D}^i)), \mathrm{bnd}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))), 0), \\ &\dots, (p^i_n, \min\{r^i_n + v_{\max}\Delta t, \mathrm{diam}(S)\}\big),\end{aligned}$$

where $\mathcal{A}^i = \{i\} \cup \mathrm{argmin}_{j \in \{1,\dots,n\} \setminus \{i\}} r^i_j$.

Acquisition of up-to-date information: In each timestep, agents are faced with the decision of whether to acquire up-to-date information about the location of other agents. This is captured by the set-valued map $\mathcal{U} : (S \times \mathbb{R}_{\geq 0})^{n^2} \rightrightarrows (S \times \mathbb{R}_{\geq 0})^{n^2}$ that, to $\mathcal{D} \in (S \times \mathbb{R}_{\geq 0})^{n^2}$, associates the Cartesian product $\mathcal{U}(\mathcal{D})$ whose ith component is either $\mathcal{D}^i$ (agent i does not get new information) or the vector

$$((p'_1, r'_1), \dots, (p'_n, r'_n))$$

where $(p'_j, r'_j) = (p^j_j, 0)$ for $j \in \{i\} \cup \mathcal{N}_i$ and $(p'_j r'_j) = (p^i_j, r^i_j)$ otherwise (agent i gets new information). Recall that $\mathcal{N}_i$ is the set of neighbors to agent i given the partition $\mathcal{V}(\mathrm{loc}(\mathcal{D}))$. It is not difficult to show that the set-valued map $\mathcal{U}$ is closed (a set-valued map $T : X \rightrightarrows Y$ is closed if $x_k \to x$, $y_k \to y$ and $y_k \in T(x_k)$ imply that $y \in T(x)$). We define the set-valued map $T : (S \times \mathbb{R}_{\geq 0})^{n^2} \rightrightarrows (S \times \mathbb{R}_{\geq 0})^{n^2}$ by $T = \mathcal{U} \circ \mathcal{M}$. Given the continuity of $\mathcal{M}$ and the closedness of $\mathcal{U}$, the map T is closed. Let $\gamma = \{\mathcal{D}(t_\ell)\}_{\ell \in \mathbb{Z}_{\geq 0}}$ be an evolution of the `self-triggered centroid algorithm`, then $\gamma' = \{\mathcal{D}'(t_\ell)\}_{\ell \in \mathbb{Z}_{\geq 0}}$, with $\mathcal{D}'(t_\ell) = f_{\mathrm{info}}(\mathcal{D}(t_\ell))$, is a trajectory of the dynamical system

$$\mathcal{D}'(t_{\ell+1}) \in T(\mathcal{D}'(t_\ell)). \tag{11}$$

With T formally defined, one can show that the aggregate function $\mathcal{H}$ is monotonically nonincreasing along the trajectories of T. Furthermore, it can also be shown that the omega limit set $\Omega(\gamma')$ is weakly positively invariant. The final step then uses this fact to show that $\Omega(\gamma')$ is contained in the set of centroidal Voronoi configurations [16].

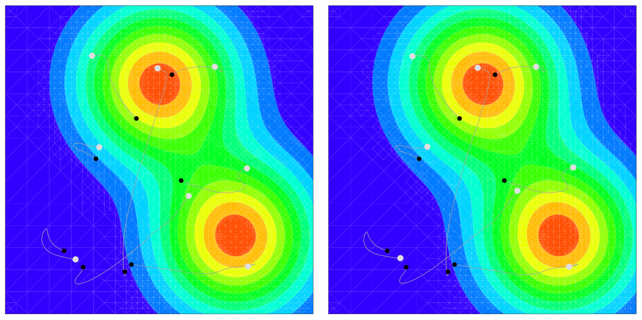
VII. Simulations

In this section, we provide several simulations of the `self-triggered centroid algorithm`. All simulations are done with $n = 8$ agents moving in a 4m × 4m square, with a maximum velocity $v_{\max} = 1$m/s operating with $\Delta t = .025$s. We compare our algorithm against the move-to-centroid strategy where agents have perfect location information at all times, see [1]; we refer to this as the benchmark case. For each agent $i \in \{1, \dots, n\}$, we adopt the following model [20] for quantifying the total power $\mathcal{P}_i$ used by agent i to communicate, in $dBmW$ power units:

$$\mathcal{P}_i = 10 \log_{10} \left[\sum_{j \in \{1,\dots,n\}, i \neq j}^{n} \beta 10^{0.1 P_{i \to j} + \alpha \|p_i - p_j\|} \right]$$

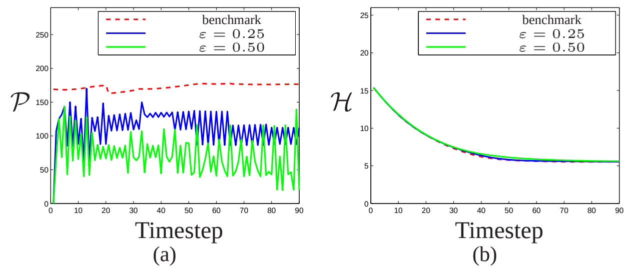
where α and β are positive real parameters that depend on the characteristics of the wireless medium and $P_{i \to j}$ is the power received by agent j of the signal transmitted by agent i. In our simulations, all these values are set to 1.

Figures 3 and 4 illustrate an execution of the `self-triggered centroid algorithm` for a density ϕ which is a sum of two Gaussian functions, and compare its performance against the benchmark case. One can see in Figure 4 how, as ε gets larger, the communication effort of the agents decreases, at the cost of a slower convergence on the value of $\mathcal{H}$.



(a) (b)

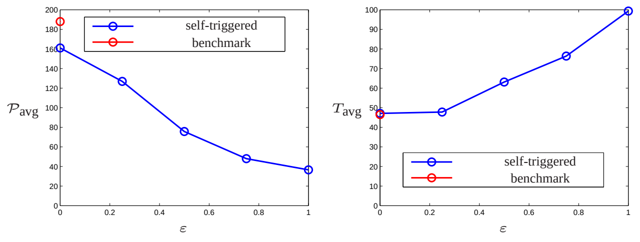
Fig. 3. Network trajectories of (a) the benchmark case and (b) the `self-triggered centroid algorithm` with $\varepsilon = 0.25$. The black and grey dots correspond to the initial and final agent positions, respectively.



(a) (b)

Fig. 4. Plots of (a) the communication power $\mathcal{P}$ used by the network and (b) the value of $\mathcal{H}$ in each timestep comparing three different executions.

Figure 5 shows the communication power used and the time to convergence of the `self-triggered centroid algorithm` averaged over 20 random initial conditions for varying ε. Note that for small ε, the network performance does not deteriorate significantly while the communication effort by the individual agents is substantially smaller.



(a) (b)

Fig. 5. Plots of the average (a) communication power consumption $\mathcal{P}_{\text{avg}}$ and (b) timesteps to convergence T_{avg} over 20 simulations for varying ε.

VIII. Conclusions

We have proposed the `self-triggered centroid algorithm`. This strategy combines an update law to determine when old information needs to be updated and a motion control law that uses this information to decide how to best move. We have analyzed the correctness of the proposed algorithm using tools from computational geometry and set-valued analysis. Our simulations have illustrated our theoretical results and have shown a performance our algorithm comparable to the constant communication, perfect information case, while requiring substantially less communication effort. In future work, we plan to characterize analytically the tradeoff between performance and communication cost, provide guarantees on the network energy savings when using the proposed algorithm, and explore the extension of these ideas to other motion coordination tasks.

Acknowledgments

This research was supported in part by NSF awards CMMI-0908508 and CCF-0917166.

References

[1] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

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

[3] A. Arsie and E. Frazzoli, "Efficient routing of multiple vehicles with no communications," *International Journal on Robust and Nonlinear Control*, vol. 18, no. 2, pp. 154–164, 2007.

[4] A. Kwok and S. Martínez, "Deployment algorithms for a power-constrained mobile sensor network," *International Journal on Robust and Nonlinear Control*, vol. 20, no. 7, pp. 725–842, 2010.

[5] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Distributed algorithms for environment partitioning in mobile robotic networks," *IEEE Transactions on Automatic Control*, 2010. To appear.

[6] M. Schwager, D. Rus, and J. J. Slotine, "Decentralized, adaptive coverage control for networked robots," *International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, 2009.

[7] C. G. Cassandras and S. Lafortune, *Introduction to Discrete-Event Systems*. Springer, 2 ed., 2007.

[8] M. Velasco, P. Marti, and J. M. Fuertes, "The self triggered task model for real-time control systems," in *Proceedings of the 24th IEEE Real-Time Systems Symposium*, pp. 67–70, 2003.

[9] R. Subramanian and F. Fekri, "Sleep scheduling and lifetime maximization in sensor networks," in *Symposium on Information Processing of Sensor Networks*, (New York, NY), pp. 218–225, 2006.

[10] X. Wang and M. D. Lemmon, "Self-triggered feedback control systems with finite-gain L2 stability," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 452–467, 2009.

[11] A. Anta and P. Tabuada, "To sample or not to sample: self-triggered control for nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2030–2042, 2010.

[12] P. Wan and M. D. Lemmon, "Event-triggered distributed optimization in sensor networks," in *Symposium on Information Processing of Sensor Networks*, (San Francisco, CA), pp. 49–60, 2009.

[13] D. V. Dimarogonas and K. H. Johansson, "Event-triggered control for multi-agent systems," in *IEEE Conf. on Decision and Control*, (Shanghai, China), pp. 7131–7136, 2009.

[14] M. Mazo Jr. and P. Tabuada, "Decentralized event-triggered control over wireless sensor/actuator networks," *IEEE Transactions on Automatic Control*, 2011. To appear.

[15] A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Event-triggered control for discrete-time systems," in *American Control Conference*, (Baltimore, MD), pp. 4719–4724, July 2010.

[16] C. Nowzari and J. Cortés, "Self-triggered coordination of robotic networks for optimal deployment," *Automatica*, 2010. Submitted.

[17] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics, Wiley, 2 ed., 2000.

[18] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: Applications and algorithms," *SIAM Review*, vol. 41, no. 4, pp. 637–676, 1999.

[19] J. Sember and W. Evans, "Guaranteed Voronoi diagrams of uncertain sites," in *Canadian Conference on Computational Geometry*, (Montreal, Canada), 2008.

[20] S. Firouzabadi, "Jointly optimal placement and power allocation in wireless networks," Master's thesis, University of Maryland at College Park, 2007.