Project Report: Self-triggered communication for higher order mobile coverage control

Venkata Sasikiran Veeramachaneni

***Abstract*— Most current results in the higher order mobile coverage control assumes perfect and instantaneous communication at all times between the neighboring agents. This is unrealistic in the present digital world. In this paper we study a higher order deployment problem for a group of agents where individual agents operate with outdated information on each others locations. Our objective is to how far we can count on the outdated information on neighboring agents and at which point it becomes essential to obtain new, up-to-date information. We introduce a self-triggered communication algorithm which guides each agent when to communicate with the neighboring agents and how to move in the absence of perfect information on the agents.**

I. Introduction

In the recent years, the technology has advanced so much that we are able to deploy large group of agents/robots/sensors to accomplish tasks like environmental monitoring [**?**],data collection [**?**] etc. The employment of teams of agents has numerous potential advantages. For instance, certain tasks may be impossible for a single agent to accomplish. This also gives robustness to the system from single agent failures. One fundamental problem in this area is 'How to place these group of agents?' The answer for the question is to place agents in an area in such a way that some predefined coverage performance function can be optimized. The performance function can be related to the sensing quality of the agents in a mobile sensing network or the probability of occurrence of events those can be sensed by the agents in the area of consideration. This problem has been studied in many papers and of them the famous one is [**?**] which characterized and optimized notions of quality-of-service provided by an adaptive sensor network.

Typical coverage problem usually considers any point in the area of interest should be monitored by any one of the agents in a group. This may not be the case in some real-world applications where, in a coverage task, more than one agent is required to monitor any point taken in the area of interest. For example, in bi-static radar deployment, it is required that the transmitter and receiver are at different locations. Though the transmitter is not a sensor, both the transmitter and receiver need to be reasonably close to a potential target in order to have satisfactory detection; thus the transmitter from coverage point of view it is like a sensor. Another application is Geo-localization using TDOA sensors. In this application atleast three sensors at different non-collinear locations are required to localize an object. This problem was first studied in [**?**] has provided framework to deal with such kind of problems. However, the framework assumed that the agents have continuous access to state information about their neighbors and can update their control signal continuously. Unfortunately this is not a fair assumption in the context of cyber-physical systems since the digital controllers cannot update their actuator signals continuously nor agents be in constant communication with one another. we want to deal with the issue by designing a naturally synchronous algorithm that endows agents with sufficient level of autonomy. This bring us the other area of relevance to this project which are works that study self-triggered decentralized strategy that are based on local interactions with neighbors in some network [**?**]. In this paper, we draw inspiration from [**?**] and try to address the above mentioned drawbacks in the higher order coverage control framework.

Statement of contributions: The main contribution of the paper is to study a robotic sensor network performing an higher order optimal static deployment task when the individual agents have no up-to-date information on the neighboring agents. We design a `self-triggered centroid algorithm` to achieve higher order optimal coverage in convex environment. We first design an update policy that helps an agent determine if the information it possess about the other agents is sufficiently up-to-date. We then focus on motion control law which determines the best motion plan based on the information an agent has to achieve higher order optimal deployment task. To execute the proposed algorithm, the individual only need to have information about a subset of the network and does not need to know the total number of agents.

Organization: Section 2 outlines some important notions from computational geometry. Section 3 talks about the problem statement. Section 4 presents our algorithm design. Section 5 assimilates the conclusions.

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

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

Let $s_1, s_2, ..., s_n$ be n partitions of S then,

$$M_S = \sum_{i=1}^{n} M_{s_i}, \qquad C_S = \frac{\sum_{i=1}^{n} M_{s_i} C_{s_i}}{\sum_{i=1}^{n} M_{s_i}}$$

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 **??** 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) $\|p - \mathrm{pr}_{\overline{B}(q,r)}(p)\| \leq \delta$.

We denote by $\overline{B}(p, r)$ the closed ball centered at $p \in S$ with radius r and by $H_{po} = \{q \in \mathbb{R}^d \mid \|q - p\| \leq \|q - o\|\}$ the closed halfspace determined by $p, o \in \mathbb{R}^d$ that contains p.

B. Voronoi partitions

We briefly present some concepts related to Voronoi partitions here. Let S be a convex polygon in $\mathbb{R}^2$ and $P = (p_1, \ldots, p_n)$ be the location of n sensors. A *partition* of S is a collection of n polygons $\mathcal{K} = \{K_1, \ldots, K_n\}$ with disjoint interiors whose union is S. The *Voronoi partition* $\mathcal{V}(P) = \{V_1, \ldots, V_n\}$ of S generated by the points $P = (p_1, \ldots, p_n)$ is

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

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 $\mathcal{N}_i$. $P = (p_1, \ldots, p_n)$ is a *centroidal Voronoi configuration* if it satisfies that $p_i = C_{V_i}$, for all $i \in \{1, \ldots, n\}$.

The k Voronoi partition of a convex area S is given below. Let P be the set of sensors positions in S. Suppose further that $\mathcal{T}$ is a subset of P and there are k elements in $\mathcal{T}$. The generalized Voronoi partition is defined as

$$V(\mathcal{T}) = \{q | \forall v \in \mathcal{T}, \forall w \in P \backslash \mathcal{T} \mid \|q, v\| \leq \|q, w\|\,,\ |\mathcal{T}| = k\}.$$

where $P\backslash\mathcal{T}$ denotes the relative component of $\mathcal{T}$ with respect to P. For each point q in $V(\mathcal{T})$, q is not further to any sensor in $\mathcal{T}$ than to any sensor not in $\mathcal{T}$. In paper we largely focus on the order 2 coverage problem. So, through out the paper $k = 2$ and $|\mathcal{T}| = 2$.

For all $p_i \in P$, there are some $\mathcal{T}$ that contain p_i. We put all these $\mathcal{T}$ into a set $\mathcal{P}_i = \{\mathcal{T} | \mathcal{T} \subset P, p_i \in \mathcal{T}\}$. Further suppose that $\mathcal{W}_i = \cup_{\mathcal{T} \in \mathcal{P}_i} V_{\mathcal{T}}$.It is noticeable that when we put these $\mathcal{T}$ together, we will not obtain the same cell containing p_i in the order 1 partition. In fact there holds $p_i \in V_i \subset W_i$.in addition, according to the definition of higher order Voronoi partition, V_i and $V_{\mathcal{T}}$ are both always convex but W_i amy not be convex.

C. Performance function and its realtionship with higher order Voronoi partition

We introduce here the performance function which needs to be minimized in order to achieve the optimal deployment of sensors/agents. Define a set $C = \{i, j | i, j \in \{1, \ldots, n\}, i < j\}$. We also note that the set designation of a particular $\mathcal{T}$ as $\mathcal{T}_{ij}$ means points $p_i, p_j \in \mathcal{T}_{ij}$ and $(i, j) \in C$. The generalized performance function can be given as follows

$$\mathcal{H}(P) = \int_S \min_{(i,j) \in C} f(\|q, p_i\|, \|q, p_j\|)\phi(q)dq \qquad (1)$$

where ϕ is the distribution density function known to all sensors and $f(.,.)$ indicates the measurement quality of a point q by a pair of agents. The function $f(.,.)$ should be differentiable and should also obey the following properties:

(i)
$$\frac{\partial}{\partial \|q, p_i\|} f(\|q, p_i\|, \|q, p_j\|) \geq 0$$

(ii)
$$\frac{\partial}{\partial \|q, p_j\|} f(\|q, p_i\|, \|q, p_j\|) \geq 0$$

(iii)
$$f(\|q, p_i\|, \|q, p_j\|) = f(\|q, p_j\|, \|q, p_i\|)$$

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

$$\mathcal{H}(P) = \sum_{\forall \mathcal{T}_{ij} \subset P}^{n} \int_{V_{\mathcal{T}_{ij}}} f(\|q, p_i\|, \|q, p_j\|)\phi(q)dq,$$

The function $\mathcal{H}$ is to be minimized with respect to both the sensors' locations P and the assignment of the Voronoi paritions $V_{\mathcal{T}_{ij}}$.

III. Problem statement

Consider a group of agents moving in a convex polygon $S \subset \mathbb{R}^2$ with positions $p_1, \ldots, p_n$. For simplicity, we consider first-order continuous-time dynamics. 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 $\ell \in \mathbb{Z}_{\geq 0}$, and
(ii) each agent can move a maximum amount of $v_{\max}$ in one second, i.e., $\|p_i(t_{\ell+1}) - p_i(t_\ell)\| \leq v_{\max}\Delta t$.

For simplicity of presentation, we consider the case of a common maximum velocity bound $v_{\max}$ for all agents.

Our objective is to achieve optimal deployment, measured according to the performance function $\mathcal{H}$ introduced in (**??**), 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_j^i and the time elapsed

$\tau_j^i \in \mathbb{R}_{\geq 0}$ 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 $r_j^i = v_{\max}\tau_j^i$ from p_j^i, and hence agent i can construct a ball $\overline{B}(p_j^i, r_j^i)$ that is guaranteed to contain the actual location of agent j. This data is stored in the vector

$$\mathcal{D}^i = ((p_1^i, r_1^i), \dots, (p_n^i, r_n^i)) \in (S \times \mathbb{R}_{\geq 0})^n.$$

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 **??**, this is not always necessary. We refer to $\mathcal{D} = (\mathcal{D}^1, \dots, \mathcal{D}^n) \in (S \times \mathbb{R}_{\geq 0})^{n^2}$ as the entire memory of the network. We find it convenient to define the map $\mathrm{loc} : (S \times \mathbb{R}_{\geq 0})^{n^2} \to S^n$ to extract the exact agents' location information from $\mathcal{D}$ by $\mathrm{loc}(\mathcal{D}) = (p_1^1, \dots, p_n^n)$.

To optimize $\mathcal{H}$, based on gradient control, the agents have to move in the direction towards the centroid of $\mathcal{W}_i$ [**?**]. So, the knowledge of the Voronoi cell and their union $\mathcal{W}_i$ is critical to each agent. However with the above data structure, agents cannot compute the Voronoi partitions exactly which in turn leads to inability to compute the exact position of centroid of $\mathcal{W}_i$. We address this issue by self-triggered approach in the next section.

IV. Self-triggered higher order coverage optimization

Here we design a coordination strategy to solve the problem described in Section **??**. 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 W_i cell and move towards its centroid, as in [**?**]. 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 the last known true positions of the other agents to compute the initial centroid of W_i cell and moves towards the centroid.

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

Lemma IV.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 **??**, if agent i is at $p = p_i$, computes the target $q = C_{W_i}^{init}$ and moves towards it to p', then the distance to $q^* = C_{W_i}^{true}$ decreases as long as

$$\|p' - C_{W_i}^{init}\| \geq \|C_{W_i}^{true} - C_{W_i}^{init}\| \tag{2}$$

holds. The right-hand side cannot be computed exactly by i because of lack of information about $C_{W_i}^{true}$ but this can be upper bounded and we show next.

Proposition IV.2 *For any density function ϕ, the following holds*

$$\|C_{W_i}^{true} - C_L^{init}\| \leq \mathrm{bnd}_i \tag{3}$$

(....This need to be proved.....)

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 (**??**) holds by making sure that

$$\|p' - C_{W_i}^{init}\| \geq \mathrm{bnd}_i \tag{4}$$

holds. The point p' to which agent i moves to is determined as follows: move towards $C_{W_i}^{init}$ 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 **??**.

Algorithm 1: `motion control law`

Agent $i \in \{1, \dots, n\}$ performs:
1: set $D = \mathcal{D}^i$
2: compute $q = C_{W_i}^{init}$ and $r = \mathrm{bnd}_i$
3: move to $\mathrm{tbb}(p_i, v_{\max}\Delta t, q, r)$
4: set $\mathcal{D}_j^i = (p_j^i, r_j^i + v_{\max}\Delta t)$
5: 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 (**??**) grows larger and (**??**) 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 **??**.

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 (**??**) is feasible. Two reasons can make (**??**) 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 $\mathcal{D}^i$. On the other hand, agent i might be close to $C_{W_i}^{init}$. Both the scenarios should trigger the need for up-to-date information through communication with other agents.

Formally, the memory updating mechanism followed by each agent is described by the pseudo-code in Algorithm **??**.

According to Algorithm **??**, agent i checks at each time step if condition (**??**) is feasible and therefore it is advantageous to execute the `motion control law` for one timestep.

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

Agent $i \in \{1, \dots, n\}$ performs:

1: set $D = \mathcal{D}^i$
2: compute $q = C^{init}_{W_i}$ and $r = \mathrm{bnd}_i$
3: **if** $r \geq \|q - p_i\|$ **then**
4: reset $\mathcal{D}^i$ by acquiring up-to-date location information
5: **end if**

C. The `self-triggered centroid algorithm`

The self-triggered coordination algorithm is the result of combining the motion control law of Section **??** and the update policy of Section **??** with a procedure to acquire up-to-date information about other agents when this requirement is triggered (cf. `4:` in Algorithm **??**). 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 W_i cell neighbors at the specific time when step `4:` is executed. We know that $W_i \supset V_i$. So, we first compute Voronoi cell. The algorithm to compute this is `Voronoi cell computation`, borrowed from [**?**]. We present it in Algorithm **??**, 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 $W(p_i, R_i) = \overline{B}(p_i, R_i) \cap \left(\cap_{j : \|p_i - p_j\| \leq R_i} H_{p_i p_j} \right)$
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 $W(p_i, R_i) = \overline{B}(p_i, R_i) \cap \left(\cap_{j : \|p_i - p_j\| \leq R_i} H_{p_i p_j} \right)$
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. Based on the Voronoi cell obtained from Algorithm **??**, we compute the W_i using W_i `cell computation` algorithm. We present it in Algorithm **??**,

Algorithm 4: W_i `cell computation`

(..... to be done yet.....)

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

V. 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.

Algorithm 5: `self-triggered centroid algorithm`

Initialization

1: set $\mathcal{D}^i$ and $\mathcal{A}^i$ by running W_i `cell computation`
2: set $\mathcal{D}^i_i = (p_i, 0)$

At timestep ℓ, agent $i \in \{1, \dots, n\}$ performs:

1: set $D = \pi_{\mathcal{A}^i}(\mathcal{D}^i)$
2: compute $q = C^{init}_{W_i}$ and $r = \mathrm{bnd}_i$
3: **if** $r \geq \|q - p_i\|$ **then**
4: reset $\mathcal{D}^i$ and $\mathcal{A}^i$ by running W_i `cell computation`
5: set $D = \pi_{\mathcal{A}^i}(\mathcal{D}^i)$
6: set $q = C^{init}_{W_i}$ and $r = \mathrm{bnd}_i$
7: **end if**
8: move to $\mathrm{tbb}(p_i, v_{\max}, q, r)$
9: set $\mathcal{D}^i_i = (\mathrm{tbb}(p_i, v_{\max}, q, r), 0)$
10: set $\mathcal{D}^i_j = (p^i_j, r^i_j + v_{\max} \Delta t)$ for $j \neq i$

References

[1] T. B. Curtin, J. G. Bellingham, J. Catipovic, and D. Webb, "Autonomous oceanographic sampling networks," *Oceanography*, vol. 6, no. 3, pp. 86–94, 1993.

[2] P. E. Rybski, N. P. Papanikolopoulos, S. A. Stoeter, D. G. Krantz, K. B. Yesin, M. Gini, R. Voyles, D. F. Hougen, B. Nelson, and M. D. Erickson, "Enlisting rangers and scouts for reconnaissance and surveillance," *IEEE Robotics & Automation Magazine*, vol. 7, no. 4, pp. 14–24, 2000.

[3] C. Nowzari and J. Cortés, "Self-triggered coordination of robotic networks for optimal deployment," *Automatica*, vol. 48, no. 6, pp. 1077–1087, 2012.

[4] J. Cortes, S. Martinez, 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.

[5] B. Jiang, Z. Sun, and B. D. Anderson, "Higher order voronoi based mobile coverage control," in *American Control Conference (ACC), 2015*. IEEE, 2015, pp. 1457–1462.