A Projected Lloyd's Algorithm for Coverage Control Problems

Yoav Palti

Faculty of Aerospace Engineering, Technion - Israel Institute Of Technology

M.Sc. Seminar

Supervisor: Associate Professor Daniel Zelazo

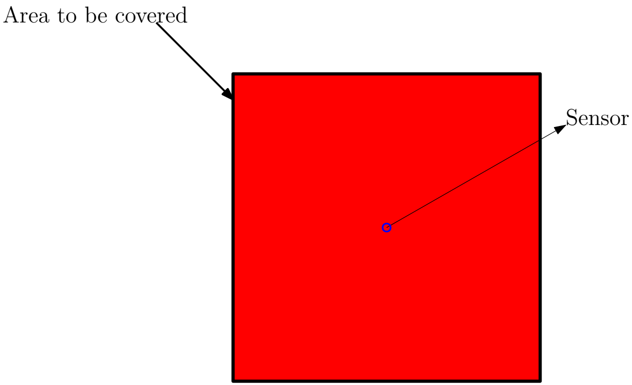
December 10, 2018

Table of Contents

Table of Contents

What Is Sensor Coverage?

Given an area, we want to sense what's happening inside



In red - the sensing coverage.

What Is Sensor Coverage?

Why would we like to do that?

- Surveillance
- Photographing
- iRobot!

Those are all sensing coverage problem[1,2,3].

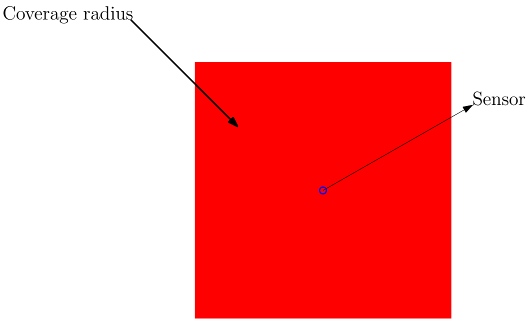
[1] Nigam, N., Bieniawski, S., Kroo, I., & Vian, J. (2012). Control of multiple UAVs for persistent surveillance: Algorithm and flight test results. IEEE Transactions on Control Systems Technology, 20(5), 1236–1251.

[2] Montijano, E., Sagues, C., & Llorente, S. (2016). Multi-Robot Persistent Coverage with Optimal Times, (Cdc), 3511–3517.

[3] Loizou, S. G., & Constantinou, C. C. (2016). Multi-Robot Coverage on Dendritic Topologies Under Communication Constraints, (Cdc).

What Is Sensor Coverage?

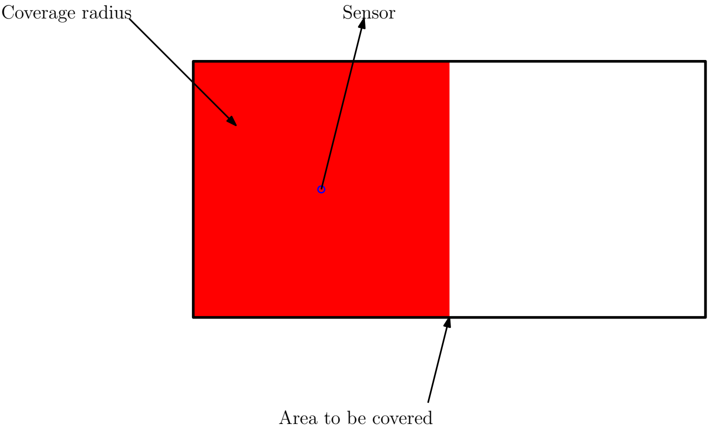
Coverage "Radius" - how much a sensor can sense



In red - the sensor coverage *radius*

Partial Coverage

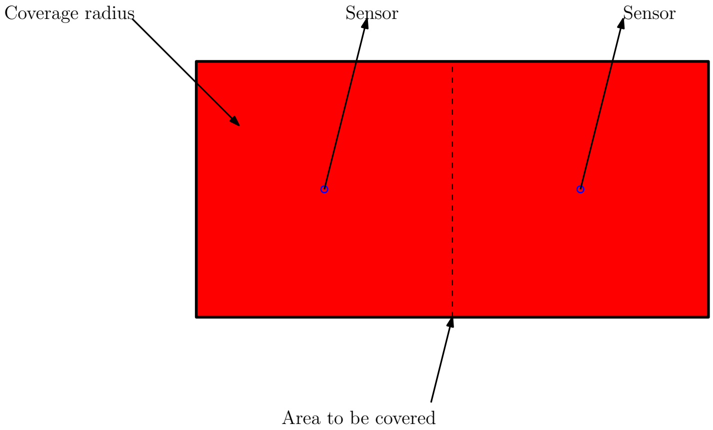
Single sensor - double the area size



In red - the sensor coverage. No *full coverage*!

Full Coverage (again)

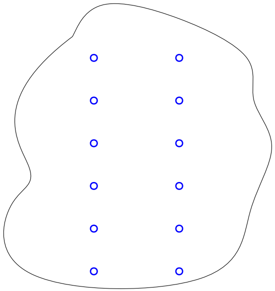
Let's add another sensor!



New sensor added - now we have *full coverage* once again!

Deployment

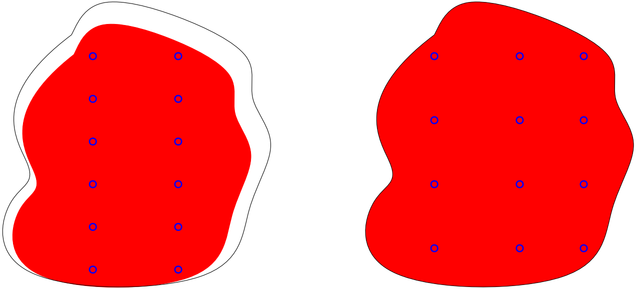
Now we're dealing with multiple sensors. How should we configure them?



We have 12 sensors which we can *deploy* in various configurations.

Deployment and full coverage

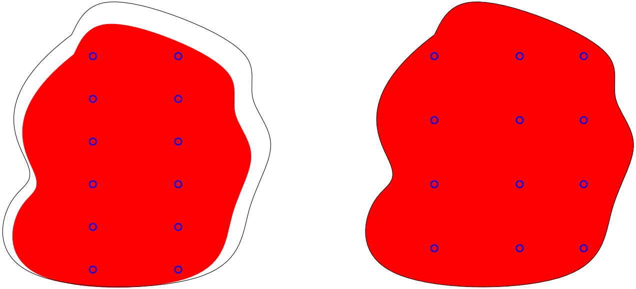
Is it that simple?



One configuration results with full coverage, while the other one doesn't.

Deployment and full coverage

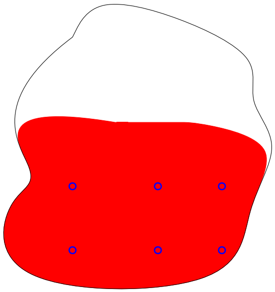
Is it that simple?



One configuration results with full coverage, while the other one doesn't.

Partial Coverage

There exists a deployment with 12 sensors which can cover the area. What if we only have 6 sensors?

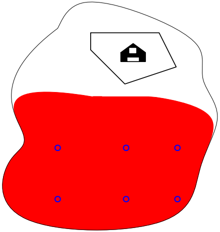


There doesn't exists a configuration that can supply full coverage!

Making it a bit more interesting...

Let's say that we want to maintain coverage on a specific area, due to:

- Connection to home base
- Maintain surveillance on a target
- ...



We have to take this into account when we build our coverage *strategy*.

Partial Coverage Strategy

Dealing with partial coverage - many possible behaviours:

- Set of trajectories [1,2,3]
- *Tiling the area*
- ...

By choosing any strategy, a *coverage controller*[4] must be provided.

[1] Atinç, G. M., Stipanović, D. M., Voulgaris, P. G., & Karkoub, M. (2013). Supervised coverage control with guaranteed collision avoidance and proximity maintenance. Proceedings of the IEEE Conference on Decision and Control, 3463–3468.

[2] Hussein, I. I., & Stipanovic, D. M. (2007). Effective Coverage Control for Mobile Sensor Networks With Guaranteed Collision Avoidance. IEEE Transactions on Control Systems Technology, 15(4), 642–657.

[3] Du, Q., Faber, V., & Gunzburger, M., "Centroidal Voronoi Tessellations: Applications and Algorithms," SIAM Review, Vol. 41, No. 4, 1999, pp. 637–676.

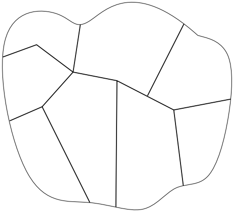
[4] Cassandras, C. G., & Li, W. (2005). Sensor Networks and Cooperative Control. Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference On, 4237–4238.

Partitioning as a strategy

Partitioning (or tiling) an area - cover a small part of the area each time.

- Main benefit - provide coverage of a subset of an area constantly.

Cortes et al.[1] provided a controller that know how to partition an area and provide coverage.



[1] Cortes, J., & Martinez, S. (2004). Coverage control for mobile sensing networks. IEEE Transactions on Robotics and Automation, 20(2), 243-255.

Problem Formulation

- There is some area $A \in \mathbb{R}^2$ That we aim to cover.
- A sub-area $A_m \subset A$ must be covered always (e.g. ground station).
- There exist a set of **mobile** sensors $S = \{s_1 \ldots s_n\}$ located in positions $p_i(t) \in \mathbb{R}^2$ (for $i = 1, \ldots, n$) at time t.
  - The sensors can be controller with integrator dynamics $\dot{p_i}(t) = u$.
  - Each sensor can cover an area defined as $Cover\,(p_i, t)$.

Problem Formulation

- A configuration c at time t is the stack of the sensor positions at time t,

$$c\left(t\right) = \begin{bmatrix} p_1^T\left(t\right) & \cdots & p_n^T\left(t\right) \end{bmatrix}^T \in \mathbb{R}^{2n}$$

  - The coverage of a configuration $D\left(c\left(t\right)\right) = \cap Cover\left(p_i, t\right)$.

Assumption

$D\left(c\left(t\right)\right) \subset A$ - a single configuration *can't* provide full coverage!

Problem Formulation

- A partition j of the area A is $pr_j \subset A$
- The partitioning of $A, PR(A)$, built from n partitions:

$$PR(A) = \{pr_j \mid \forall i \neq j,\ pr_i \cap pr_j = \emptyset \textbf{ and } \cup pr_j = A\}$$

Problem Formulation

Problem

1. *Find partitioning such that for each partition* $j, pr_j \cap A_m \neq \emptyset$
2. *Find a deployment such that for each partition* j*, and some given time* t, $A_{pr_j} \subseteq D\left(c\left(t\right)\right)$

Table of Contents

Lyapunov Stability

- Lyapunov stable - if we are neat the equilibrium poing x_eq, then the controller will stay near x_eq forever.

Lyapunov Stability

- Lyapunov stable - if we are neat the equilibrium poing x_eq, then the controller will stay near x_eq forever.
- Asymptotically stable - Lyapunov stable $+$ converge to x_eq.

Lyapunov Stability

How to prove Lyapunov stable and asymptotically stable? Using Lyapunov direct method!

1. Define a *Candidate Lyapunov Function* $V(x) : \mathbb{R}^n \to \mathbb{R}$

Lyapunov Stability

How to prove Lyapunov stable and asymptotically stable? Using Lyapunov direct method!

1. Define a *Candidate Lyapunov Function* $V(x) : \mathbb{R}^n \to \mathbb{R}$
2. To show that we're Lyapunov stable, make sure that:
   1. $V(0) = 0$
   2. $V(\zeta) > 0 \Leftrightarrow \zeta \neq 0$
   3. $\dot{V}(\zeta) \leq 0 \Leftrightarrow \zeta \neq 0$

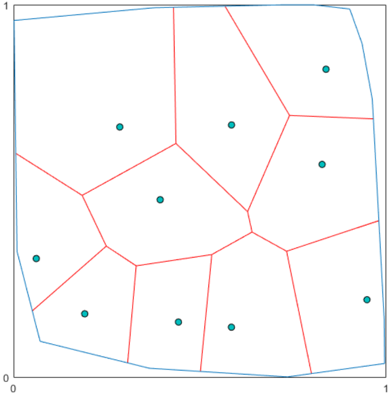
Lyapunov Stability

How to prove Lyapunov stable and asymptotically stable? Using Lyapunov direct method!

1. Define a *Candidate Lyapunov Function* $V(x) : \mathbb{R}^n \to \mathbb{R}$
2. To show that we're Lyapunov stable, make sure that:
   1. $V(0) = 0$
   2. $V(\zeta) > 0 \Leftrightarrow \zeta \neq 0$
   3. $\dot{V}(\zeta) \leq 0 \Leftrightarrow \zeta \neq 0$
3. To show that we are asymptotically stable, show that condition 2.3 is: $\dot{V}(\zeta) < 0 \Leftrightarrow \zeta \neq 0$

Voronoi Partitioning

Let's start with a simple intuitive explanation...



Voronoi Partitioning

While being a method to partition an area with some cost function, the is a widely-used representation in the coverage problem ([Cortes2004][Hussein2007][Du1999]).

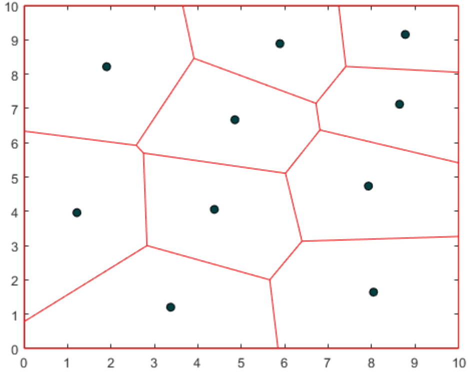
The Voronoi Diagram of a region $\Omega \subset \mathbb{R}^2$ is the set of partitions $\mathcal{V} = \{V_i \mid \cup V_i = \Omega\}$, generated by the generators $\mathcal{Z} = \{z_1, \ldots, z_n \mid z_i \in \Omega\}$, such that

$$V_i = \{q \in \Omega \mid \|q - z_i\| \leq \|q - z_j\| \forall z_i, z_j \in \mathcal{Z}\}, \tag{1}$$

where V_i corresponds to the i-th element of $\mathcal{Z}$, and $\|\cdot\|$ denotes the Euclidean distance.

Central Voronoi Tessellations

And yet again, let's have an intuitive explanation...



Central Voronoi Tessellations

Let us define a density function, ρ_i, for each Voronoi partition V_i.

Then, we can define the center of mass for each partition as

$$z_i^* = \frac{\int_{V_i} y\rho(y)dy}{\int_{V_i} \rho(y)dy}. \tag{2}$$

If a generator $z_i = z_i^* \,\forall\, V_i$, we call this partitioning a *centroidal Voronoi tessellation* (CVT).

Lloyd's Algorithm

Algorithm 1 Lloyd's Algorithm [1].

1: Calculate the Voronoi diagram for the current agents positions.
2: Calculate the center of mass for every cell.
3: Move the agents to the center of mass.

[1] Lloyd, S., "Least squares quantization in PCM," IEEE Transactions on Information Theory, Information Theory, IEEE Transactions on, IEEE Trans. Inform. Theory, Vol. 28, No. 2, 1982, pp. 129–137, doi:10.1109/TIT.1982.1056489

Lloyd's Algorithm

According to [Cortes2004], if we define agent i position as p_i and the i's partition centroid as C_{V_i}, then for some proportional constant k_{prop}, the controller can be defined as:

$$u_i = -k_p \left(p_i - C_{V_i}\right) \tag{3}$$

Moreover, this controller is locally asymptotically stable.

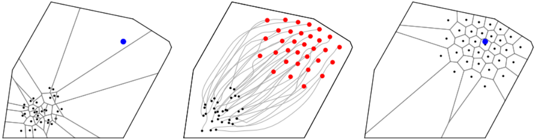


Figure: A simulation from [Cortes2004] with 32 agents

Formation Control

For a complete background, some prior knowledge on algebraic graph theory is needed. Let's simplify:

- We have agents $1 \ldots n$ on positions p_i

Formation Control

For a complete background, some prior knowledge on algebraic graph theory is needed. Let's simplify:

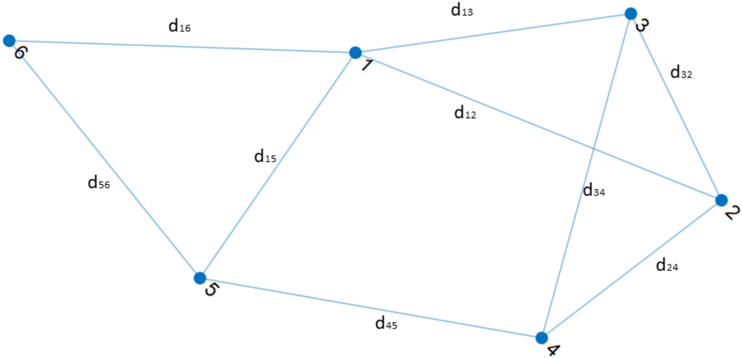
- We have agents $1 \dots n$ on positions p_i
- Goal - agents $i, j (i \neq j)$ will be at distance d_{ij}

Formation Control

For a complete background, some prior knowledge on algebraic graph theory is needed. Let's simplify:

- We have agents $1 \dots n$ on positions p_i
- Goal - agents $i, j (i \neq j)$ will be at distance d_{ij}
- Only ε agents can share information.

Formation Control



Formation Control

For a single agent p_i, the controller will have the following form:

$$\dot{p}_i = -\sum_{i \sim j} \left(\|p_i - p_j\|^2 - d_{ij}^2 \right) (p_i - p_j) \tag{4}$$

This controller is locally asymptotically stable.

Projection Operator

The projection linear operator is defined as a linear transformation P from a vector space to itself such as $P^2 = P$. In other words, the transformation P is idempotent.

Table of Contents

Projected Lloyd's Algorithm

We should supply a solution for the "Coverage Constraint"

Projected Lloyd's Algorithm

We should supply a solution for the "Coverage Constraint"

Reminder

A given area inside the area $A_m \subset A$ must be covered always

Projected Lloyd's Algorithm

We should supply a solution for the "Coverage Constraint"

Reminder

A given area inside the area $A_m \subset A$ must be covered always

We came up with a rather simple solution for this problem.

Projected Lloyd's Algorithm

Algorithm 2 Projected Lloyd's Algorithm (PLA)

1: Calculate the Voronoi diagram for the current agents positions.
2: Calculate the center of mass for every cell.
3: Project the center of mass of every cell to the area constraint limiting polygon.
4: Move the agents the projected center of mass.
5: Repeat until converge.

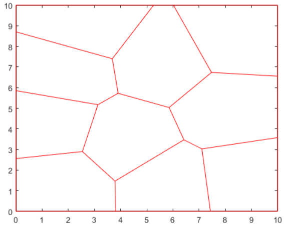
Writing this algorithm as a controller:

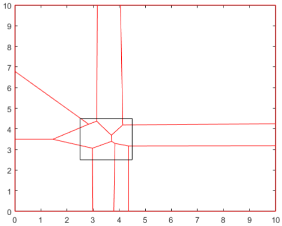
$$u_i = -k_p \left(p_i - \textit{proj} \left(C_{V_i} \right) \right) \tag{5}$$

Projected Lloyd's Algorithm

In the following example:

- Left - regular CVT, build with the original Lloyd's Algorithm.
- Right - A partitioning built with the PLA.





Projected Lloyd's Algorithm

Theorem

The projected Lloyd's Algorithm is locally asymptotically stable

As the projection is a linear operator, the controller is also locally asymptotically stable, and the proof is virtually the same as was given by Cortes et al. The proof is based on a proposal of Lyaponov function, and then using the direct Lyaponov method to prove the stability

Proof of stability

As the projection is a linear operator, the controller is also locally asymptotically stable, and the proof is virtually the same as was given by Cortes et al.

Problem Solution Algorithm

So far, we've given solution for:

- Covering a given area using Voronoi partitioning
- Partition and area such that the coverage constraint is fulfilled.

Therefore, we are ready for the problem solution algorithm...

Problem Solution Algorithm

Algorithm 3 Problem Solution Algorithm

1: Using some random initial guess, partition the whole area using PLA.

2: For each partition (assuming that the agents can actually cover each partition with their coverage radius), calculate the CVT. The initial positions for the CVT calculation is the previous partition CVT.

Lloyd's Algorithm and Formation Control

- Problem solved.

Lloyd's Algorithm and Formation Control

- Problem solved.
- Make it more interesting...

Lloyd's Algorithm and Formation Control

- Problem solved.
- Make it more interesting...
- Combine Lloyd's Algorithm with distance-based formation control!

Lloyd's Algorithm and Formation Control

- Problem solved.
- Make it more interesting...
- Combine Lloyd's Algorithm with distance-based formation control!
  - create and maintain spatial properties partially or fully (We do not provide a condition where this combination meets the requirements).

Lloyd's Algorithm and Formation Control

As both of the controllers are convex, we propose to simply combine them with some coefficient:

$$u_i = \alpha\left(-k_p\left(p_i - C_{V_i}\right)\right) + (1-\alpha)\left[-\sum_{i\sim j}\left(\|p_i - p_j\|^2 - d_{ij}^2\right)\left(p_i - p_j\right)\right] \quad (6)$$

Lloyd's Algorithm and Formation Control

As both of the controllers are convex, we propose to simply combine them with some coefficient:

$$u_i = \alpha \left(-k_p \left(p_i - C_{V_i}\right)\right) + (1-\alpha)\left[-\sum_{i \sim j}\left(\|p_i - p_j\|^2 - d_{ij}^2\right)\left(p_i - p_j\right)\right] \quad (6)$$

Theorem

The combined controller is Locally Asymptotically Stable

Lloyd's Algorithm and Formation Control

How to prove:

- Pretty long and technical, based on Lyapunov function.
- We know the Lyapunov function of each controller - Let's combine!
- After long calculations, we can show using the Lyapunov direct method that this controller is locally asymptotically stable.
- In the same way, we can show it works with the PLA.

Table of Contents

Some Simulation

List of simulations to create:

- 3 agents, 5 big partitions, no formation, no PLA
- 3 agents, 5 big partitions, no formation, PLA
- 10 agents, 5 big partitions, no formation, PLA
- 6 agents, 5 big partitions, some formation, PLA