Coverage control

Marta Arriaza, Francisco del Campo and Carla Lázaro
Physics Engineering Project II
ETSETB- UPC
(Dated: June 4, 2021)

The focus of this project is on optimal coverage of a given geographical area by means of sensors. To this aim, we reproduce a dynamic method to locate the sensors. The implementation is made by gradient descent-based control algorithms relying upon computational geometry of spatial structures, in particular, Voronoi diagrams.

I. INTRODUCTION

Motivation: The focus of this project is the design of control and coordination algorithms for groups of vehicles that perform sensing tasks. We refer to the groups of vehicles as sensor networks. These networks can perform several relevant actions, that are difficult to carry out for a single sensor, such as search and recovery operations, manipulation in hazardous environments, exploration, surveillance and environmental monitoring for pollution detection and estimation.

Sharing information between the vehicles of the network, provides the ability to adapt the system to the environment in space and time. To that end, it is important to design a control and formation architecture.

Problem description: Optimal deployment of a group of agents within a certain region from a certain initial position.

In this project, we present and implement a classical coverage control solution based on the so called Lloyd's algorithm; see [4] for a reprint of the original report and [5, 6] for numerous applications in other technological areas. Essentially, the idea is that the sensors, initially located at random positions within the target region, evolve to final positions that provide an optimal coverage of the region following a certain control strategy.

Paper organization: Section II provides a theoretical review of the mathematical concepts that are used in the implementation of the algorithm. Section III has the algorithm build up and all the main function codes and their utility in our algorithm. Finally, section IV includes several figures that show the simulation results of implementing the algorithm for a random distribution of sensors. The implementation is made varying the density function, the number of sensors and the polygon geometry.

II. LOCATION OPTIMIZATION

Let Q be a convex polygon in $\mathbb{R}^2$. We consider a *distribution density function* : $\phi : Q \to \mathbb{R}_+$ that represents the probability that some event take place over Q. Let $P = (p_1, .., p_n)$ be the location of n *sensors* moving in Q. A *partition* of Q is a collection of n polygons $W = W_1, ..., W_n$ whose union is Q. The objective is to minimize the locational optimization function:

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

A. Voronoi Partitions

At fixed sensors location, the optimal partition of Q is the Voronoi partition $V(P) = V_1, ..., V_n$ generated by the points $(p_1, p_2, ..., p_n)$:

$$V_i = \{q \in Q|\ ||q - p_i|| \leq ||q - p_j||, \forall j \neq i\} \tag{2}$$

Since Q is a convex polyhedron in a finite dimensional Euclidean space, the boundary of each V_i is a convex polygon.

In what follows, we will write: $H_v = H(P, V(P))$

B. Centroidal Voronoi Partitions

Some basic quantities associated to a region $V \subset \mathbb{R}^N$ and a mass density function ϕ are: the generalized mass and centroid (center of mass):

$$M_V = \int_V \phi(q)dq \tag{3}$$

$$C_V = \frac{1}{M_V} \int_V q\phi(q)dq, \tag{4}$$

Lets recall the locational optimization problem (1). It can be proved [1] that the critical points for H_v are *centroids* of the Voronoi cells:

$$C_{V_i} = argmin_{p_i} H_v(P)$$

C. Continuous- time Lloyd descent for coverage control

The proposed Lloyd algorithm is a *gradient descent flow*. Let $p(t) \in \mathbb{R}^2$ denote the position of the *ith* vehicle at time t. Assume the sensors location obeys the

following behavour:

$$\dot{p_i}(t) = u_i \tag{5}$$

We consider H_V a cost function to be minimized and impose that p_i follows a gradient descent,

$$u_i = -k(p_i - C_{V_i}) \tag{6}$$

where k is a positive gain and we assume that the partition $V(P)$ is continuously updated.

The equation (6) induces a a closed loop where the sensor location converges asymptotically to a critical point of the cost function H_V, i.e. to the centroid of the *ith* Voronoi cell [1]. With this gradient descent, it is only guaranteed to find local minima, we may not reach the global minimum.

III. ALGORITHM IMPLEMENTATION

To build the code we took some intermediate steps. Firstly, we developed the algorithm from a fixed uniform density with tabulated C_{V_i} and M_{V_i}, taking the formulas in [2]. Then we ended up applying arbitrary densities using a double integral function to calculate C_{V_i} and M_{V_i} at each iteration of the Lloyd's algorithm. All of the steps are shown in the subsections below.

A. Symmetric points

We define a region Q that contains n agents located in random locations inside it. In order to represent the Voronoi diagram inside Q, we need to calculate the sensors' position symmetric points.

The code function *simet* calculates the symmetric of a point respect the sides of Q. To compute the symmetric points, the function uses simple geometry relations to build straight lines and calculate the symmetric distances.

B. Centroids for a uniform density function

After obtaining the Voronoi diagram of our polygon Q, we used the formulas in [2] to calculate the C_{V_i} and the M_{V_i} of each Voronoi cell with uniform density distribution.

$$M_{V_i} = \frac{1}{2}\sum_{k=0}^{N_i-1}(x_k y_{k+1} - x_{k+1}y_k) \tag{7}$$

$$C_{V_{i,x}} = \frac{1}{6M_{V_i}}\sum_{k=0}^{N_i-1}(x_k + x_{k+1})(x_k y_{k+1} - x_{k+1}y_k) \tag{8}$$

$$C_{V_{i,y}} = \frac{1}{6M_{V_i}}\sum_{k=0}^{N_i-1}(y_k + y_{k+1})(x_k y_{k+1} - x_{k+1}y_k) \tag{9}$$

C. Centroids for a generic density function

For an arbitrary density function we compute the integrals in (3) and (4) for the calculation of the centroids using the *doubleintegral* function. We detail a scheme of the code below.

Name: centroid **Goal:** calculate the centroids for each Voronoi cell **Requires:** (i) Voronoi cell computation (ii) Cell's vertices (iii) Density function (iv) Doubleintegral function
1: Set the domain of integration = Voronoi cell 2: Compute Mv with doubleintegral function 3: Compute the double integral for the calculation of the centroids for both x and y component (Cvxi and Cvyi) 4: Compute the centroids as Cvx=Cvxi/Mv and Cvy=Cvyi/Mv

D. Lloyd algorithm

The implementation of Lloyd's algorithm needs the sensors position at the Voronoi cells centroid at each iteration. For this purpose, we have built the function *simncen* described below.

Name: simcen **Goal:** gives only the position of the sensors and its centroids that are in the polygon **Requires:** (i) Position of the sensor (ii) Polygon vertices (iii) Density function (iv) Simet function (v) Inpolygon function (vi) Centroid function
1: Run simet 2: Set a vector with all positions and its symmetric points 3: Compute the Voronoi regions 4: Obtain the vertices of the Voronoi cells 5: **if** vertices are inside the polygon run centoridd **end if** 6: Run inpolygon for the centroids **if in==1** Ccvx=[Ccvx Cvx] Ccvy=[Ccvy Cvy] **end if**

This function uses the symmetric points function detailed in *Subsection A. Symmetric points* and the Matlab *voronoi* function to obtain the cells vertices. It also requires the Matlab *inpolygon* function which returns 1 if the given point is situated inside the polygon described or 0 if not.

First of all, we define our control function u, as shown in (6). Then, using a Runge-Kutta method, we calculate its optimal distribution for every iteration, taking into account the density function.

Name: lloyd2F **Goal:** optimal distribution of sensor using Lloyd's algorithm **Requires:** (i) initial sensor's position (ii) Polygon's vertices (iii) Density function (iv) u function
for n=1:iterations 1: Run simcen 2: Calculate error between position and centorid 3: Run rk4 with u function 4: Plot Voronoi distribution **end for**

IV. SIMULATION RESULTS

In this section, we will show the results of applying the Lloyd algorithm, explained in the previous section. In the figures below, the red circles will represent the positions of the sensors and the blue crosses will denote the centroids of each Voronoi cell.

The algorithm has been applied to different systems, in the following figures we will see how it works modifying the density function, the number of sensors or the polygon of interest.

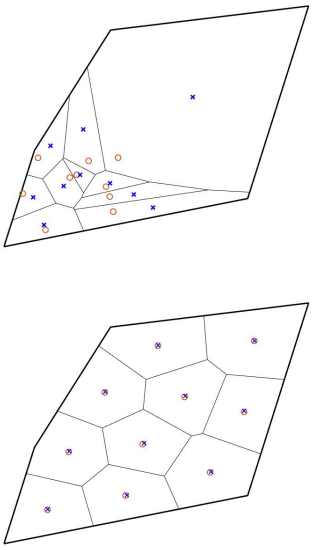


FIG. 1: Lloyd algorithm with uniform density function $\phi = 1$.

Figure 1 shows the Lloyd algorithm on a convex polygonal environment, with uniform density function $\phi = 1$. The top figure illustrates the initial position and the Voronoi cells calculated from the initial distribution. On the bottom figure, we observe the final positions of the sensors, which coincide with the centroids of each Voronoi cell for the final distribution.

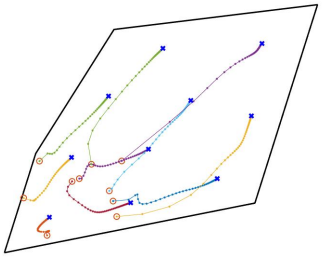


FIG. 2: Gradient descent flow for uniform density function.

Figure 2 shows the position of the vehicles after each iteration, until the position coincides with the centroid of the Voronoi cell.

The next figures show systems with a non-uniform density function. For example, on figure 3 and figure 4 we can see the behavior of the algorithm for $\phi = e^{\frac{1}{2}*(x^2+y^2)}$. The final result is similar to the uniform distribution case, then we can conclude the algorithm also works well with non-uniform density functions. To verify this statement, we see another example in *Figure 5* and *Figure 6*.

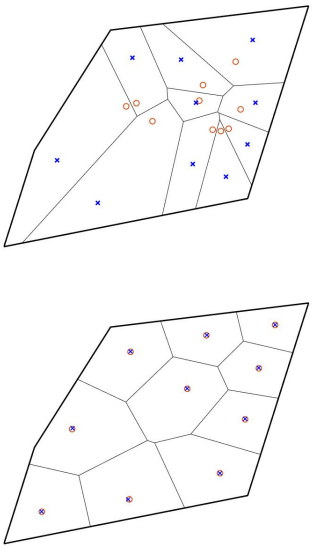


FIG. 3: Initial and final positions and Voronoi cells of the sensors distributed with $\phi = e^{\frac{1}{2}*(x^2+y^2)}$

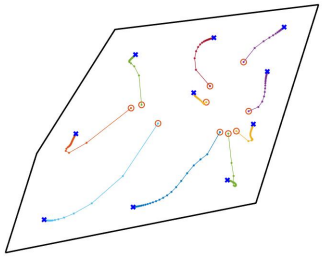
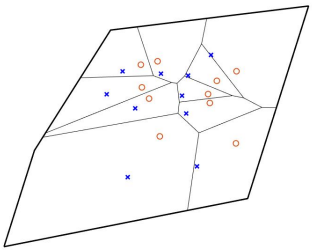


FIG. 4: Gradient descent flow for $\phi = e^{\frac{1}{2}*(x^2+y^2)}$



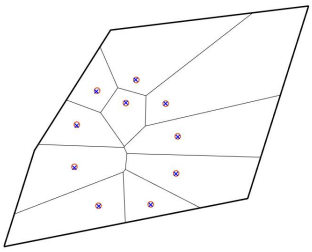


FIG. 5: Initial and final positions and Voronoi cells of the sensors distributed with $\phi = e^{-50(1.4x^2+0.6y^2-0.3)^2}$

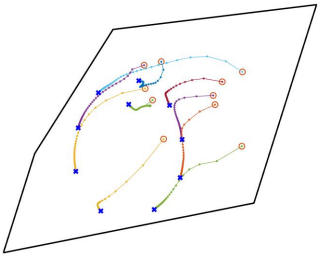
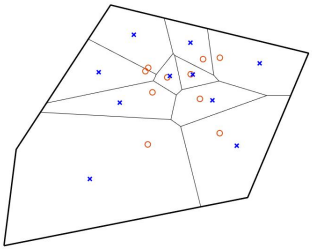


FIG. 6: Gradient descent flow for the density function $\phi = e^{-50(1.4x^2+0.6y^2-0.3)^2}$

As explained in section II, the region of interest Q has to be a convex polygon, but it can take any form. In *Figure 7* we see how the algorithm works for a different polygon if it verifies the convex condition.



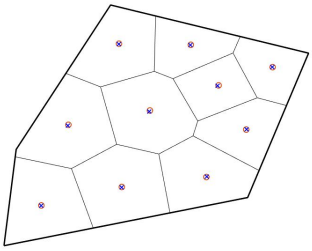
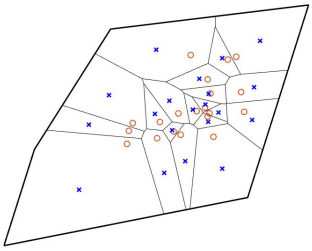


FIG. 7: Initial and final positions and Voronoi cells of the sensors distributed with $\phi = e^{\frac{1}{2}*(x^2+y^2)}$

Finally, in *Figure 8* the number of sensors is increased from *n=10* to *n=20*, we see how the algorithm gets to the same result. The number of sensors can be increased, but the computational cost increases with it.



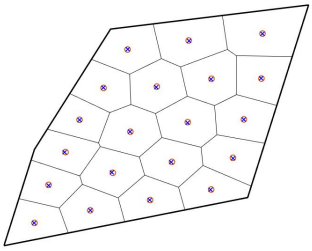


FIG. 8: Initial and final positions and Voronoi cells of the $n = 20$ sensors distributed with $\phi = 0.8((x-2)^2+(y-3)^2$

[1] Jorge Cortés, Sonia Martínez, Timur Karatas and Francesco Bullo. *Coverage Control for mobile sensing networks: variations on a theme*. Paper to IEEE transactions on Robotics and Automation (16 Dec 2002)

[2] Jorge Cortés, Sonia Martínez, Timur Karatas and Francesco Bullo. *Coverage control for mobile sensing networks*. IEEE International Conference on Robotics and Automation (May 2002)

[3] Jorge Cortes, Sonia Martínez, Timur Karatas and Francesco Bullo. *Coverage control for mobile sensing networks*. IEEE Transactions on Robotics and Automation (Volume:20, Issue:2, April 2004)

[4] S.P.Lloyd, *Least squares quantization in PCM*. IEEE Transactions on Information Theory, (Vol.28, no.2, pp.129–137, 1982)

[5] Q. Du, V. Faber, and M. Gunzburger. *"Centroidal Voronoi tessellations: applications and algorithms,"*.SIAM Review, vol. 41, no. 4, pp. 637-676 (1999).

[6] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics. John Wiley and Sons, New York, NY, second edition, (2000)