Distributed Coverage Control with Sensory Feedback for a Decentralized Multi Robot System

AKASH REDDY ALLA (S1) (1221694552) (aalla@asu.edu)

SASI VADHAN MANNE (S2) (1220393655) (smanne3@asu.edu)

Abstract: (S2) This Paper presents a control strategy that allows a group of mobile robots to position themselves to optimize the measurement of sensory information in the environment. The robots use sensed information to estimate a function indicating the relative importance of different areas in the environment, their estimate is then used to drive the network to desirable placement configuration using a computationally simply decentralized control law. We formulate the problem, provide a practical control solution, and present the results of numerical simulations. We then discuss experiments carried out on a swarm of mobile robots. We present a centralized control law for producing increased robot density in areas of high importance and decreased density in areas of low importance, specifically we consider a group of robots that is dispatched over a bounded region of interest. The groups task is to sample a sensory function over the region. The sensory function is scalar function unknown to the robots that indicates the relative importance of different areas in the region. Our solution composes an approximation of this function from sensory measurements. A centralized control law then uses this approximation, as well as neighbour positions, to drive the robots to configuration such that the sampling of the sensory function is near-optimal. This enables the network to record observations about the environment with varying resolution, so that the areas with high values of the sensory function receive higher density rate of observation than areas with low values.

Mathematical Model: (S1, S2 – Ref 2, 3 and 4)

In this section, we build a function representing the sensing cost associated with a network configuration. A network is said to have optimal coverage if it minimalizes this cost function over the region of interest. Following standard results in the field, we show that all configurations of a certain type, namely centroidal Voronoi configurations, correspond to local minima of the cost function. The sensor network consists of a group of identical robots, each with some degree of mobility and the capacity for measuring a sensory function from the environment. The sensory function indicates the relative importance of different areas in the environment, it may be quantity that is sensed directly, such as temperature, or may be demand more elaborate processing of sensory data, such as would be required to detect the concentration of a chemical, or the intensity of sound of a particular frequency 1. In addition, we assume that the robot can measure the positions of is Voronoi neighbours relative to itself and that it can detect the boundaries of the region of interest. We model the scenario described above as follows.

Let there be n robots in know, convex polytope Q$\subset R^N$ An arbitrary in Q is denoted q, the position of the i^{th} robot is denoted p_i, and the set of all robot positions is denoted P = $\{p_{1,} \dots \dots, p_n\}$. let W = $\{W_{1,} \dots \dots W_n\}$ be a partition of Q such that the one robot at position p_i lies with in each region W_i , Define the sensory function ∅(q), which is not known by the robots in the network. let the unreliability of the sensor measurement be defined by a function f(x) which is strictly increasing, f ($\|q - p_i\|$) describes how realiable is the measurements of the information at q by a sensor at p_i ($\|.\|$ Is used to denote the l^2-norm). This form of f(x) is used as a tuneable parameter to influence the sensitivity of the robots to their sensor readings

We can formulate the cost incurred by one robot sensing over one region W_i as

$$h_i(p_i, W_i) = \int f(\|q - p_i\|)\, \emptyset(q) dq. \text{ With respect to } W_i$$

Notice that unreliable sensing is expensive and high values of $\emptyset(q)$ are also expensive. summing over all robots, a function representing the overall sensing cost of a given network configuration can be written as:

$$H(P_i, W) = \sum_{i=1}^{n} \int f(\|q - p_i\|)\emptyset(q) dq. \quad (1)$$

An optimal network configuration corresponds to the particular combination of (P, W) which minimizes (1)

To solve this minimization problem, we must introduce the notion of a Voronoi partition. The Voronoi region, V_i of a given robot is the region of points that are closer to that robot than to any other, that is

$$V_i = \{q \epsilon Q \big| \|q - p_i\| \leq \|q - p_j\|, \forall j \neq i\}$$

The division of an area into such regions is called Voronoi partition, denoted V(P), and is a function of the robot positions. We will use the shorthand $H_v(P)$ =H (P, V(P)). Because the f(x) is strictly increasing and the Voronoi partition, V, minimizes the cost function, H, for any fixed configuration, P, of the robots. This is clear since, for an arbitrary point q ϵ Q, q ϵ V, we get the smallest value of f($\|q - p_i\|$) over I, and therefore the smallest contribution to H . Thus, we have

$$\text{Min } H(P, W) = min H_v$$

To find local minima of H_v, we examine solutions to the expression

$$\nabla H_v = \left[\dots \cdot \frac{\partial {H_v}^T}{\partial p_i}\right]^T = 0$$

It is clear that each partial derivative must be zero for a local minimum. Applying a multi-variable generalization of Leibniz rule, we can move the differentiation inside the integral sign in to get

$$\frac{\partial H_v}{\partial p_i} = \int_{v_i} \frac{f(\|q-p_i\|)}{\partial p_i} \emptyset(q) dq + \sum_{j \varepsilon N_i} \int_{\partial v_j} f(\|q - p_i\|)\emptyset(q) \frac{\partial \partial V_j}{\partial p_i} n_{jdq}$$

$$+ \int_{\partial v_i} f(\|q - p_i\|)\emptyset(q) \frac{\partial \partial V_i}{\partial p_i} n_{idq} \quad (2)$$

Where ∂V_i denotes the boundary of the region V_i , $n_i(q)$ denotes the outward facing normal of ∂V_i and N_i is the set of indices of the neighbours of p_i excluding itself. Note that all the

integrals in (2) are (N X 1) vectors since $\frac{\partial \partial V_j}{\partial p_i}$ is an (N X N) matrix and n_i is an (N X 1) vector. We assert that the last two terms in (2), in fact, sum to zero (negligible). A proof can be outlined as follows: since p_i only effects ∂V_j at the shared boundary of V_j & V_i, we have

$$\cup_{j \varepsilon N_i} \partial V_j = \partial V_i$$

Also, an inward normal, $-n_i$, for V_i is equal to an outward normal, n_j,for any of its neighbours V_j, at the boundary which they share. This leads to giving the desired result:

$$\sum_{j \varepsilon N_i} \int_{\partial v_j} f(\|q - p_i\|) \emptyset(q) \frac{\partial \partial V_j}{\partial p_i} n_{jdq} = -\int_{\partial v_i} f(\|q - p_i\|) \emptyset(q) \frac{\partial \partial V_i}{\partial p_i} n_{idq}$$

This procedure is known in fluid mechanics as the Reynolds transport theorem.

Using this fact, (2) can simply be written as:

$$\frac{\partial H_v}{\partial p_i} = \int_{v_i} \frac{\partial f(\|q - p_i\|)}{\partial p_i} \emptyset(q) dq$$

We can evaluate the partial derivative of f(x) using the chain rule, and move p_i outside of the integral to give:

$$\frac{\partial H_v}{\partial p_i} = \int_{v_i} \frac{q}{\|q - p_i\|_i} \frac{df(x)}{dx} |\|q - p_i\| \emptyset(q) dq + p_i \int_{v_i} \frac{1}{\|q - p_i\|_i} \frac{df(x)}{dx} |\|q - p_i\| \emptyset(q) dq \quad (3)$$

Next, we define two properties analogous to mass- moments of rigid bodies. The mass M_{v_i} of a Voronoi region v_i is defined as:

$$M_{v_i} = \int_{v_i} \frac{1}{\|q - p_i\|_i} \frac{df(x)}{dx} |\|q - p_i\| \emptyset(q) dq \quad (4)$$

And centroid of v_i is defined as

$$C_{v_i} = \frac{1}{M_{v_i}} \int_{v_i} \frac{1}{\|q - p_i\|_i} \frac{df(x)}{dx} |\|q - p_i\| \emptyset(q) dq \quad (5)$$

Note that f(x) strictly increasing and $\emptyset(q)$ is strictly positive which imply that:

$$M_{v_i} > 0 \ \& \ \forall \ v_i \neq \{0\}$$

Therefore M_{v_i} and C_{v_i} have properties intrinsic to physical masses and centroids. Substituting these quantities into (3) gives

$$\frac{\partial H_v}{\partial p_i} = -M_{v_i}(C_{v_i} - p_i) \quad (6)$$

Equation 6 implies that local minima H_v and therefore H (P,W), correspond to a configuration of P such that $p_i = C_{v_i}$, i.e; each robot is located at the centroid of its Voronoi region. We will denote the set of all such centroidal Voronoi configurations as P_C thus the optimal coverage is to drive the group of robots to a centroidal Voronoi configuration.

Theoretical Analysis: (S1, S2 – Ref 3, 6 and 7)

We will design a control law to take advantage of the surprisingly simple result in (6). The control law will drive the network to an estimated centroidal Voronoi configuration using sensory data available to the robots to form an on-line approximation of the centroids of their Voronoi regions.

Assume that the dynamics of each robot can be modelled by first order equation

$$\dot{p}_i = u_i, \qquad (7)$$

Where u_i is the control input. This simply means that a low-level controller is in place to enforce first-order dynamics. Next, we prescribe the linear proportional control law as:

$$u_i = k\,(\widehat{C_{v_i}} - p_i) \qquad (8)$$

Where $\widehat{C_{v_i}}$ is an estimate of C_{v_i} , based on the information available to robot i, to investigate the stability of such a control law we propose to use $H_v(P)$ as a Lyapunov – like function . taking the time derivative of $H_v(P)$ trajectories of (7) gives

$$\dot{H}_v = \sum_i \frac{\partial H_v}{\partial p_i}^T \dot{p}_i,$$

And using (6) and (8) we get

$$\dot{H}_v = -k\sum_{i=1}^{n} M_{v_i} e_i^T \hat{e}_i \ , \qquad (9)$$

Where $e_i = (C_{v_i} - p_i)$, and $\hat{e}_i = (\widehat{C_{v_i}} - p_i)$, the actual error, e_i is unknown. Notice, however that if an estimate, $\hat{e}_i$, can be found such that the inner product of the two errors is positive , convergence of $\hat{e}_i$ to zero will be guaranteed. Geometrically, this means that the angle between $\hat{e}_i$ and e_i must remain less than $\pi/2$ radians all time. We use this insight to design a centroid estimate, $\widehat{C_{v_i}}$ by using only sensed information local to robot i

Henceforth we will deal with a specific case in which f(x) = $1/2x^2$, this causes the factor

$$\frac{1}{\|q-p_i\|}\frac{df(X)}{dx}\,|\|q - p_i\|$$

to become unity, making the proceeding expression transparent. The method presented, however, is valid for any strictly increasing f(x) with smooth first derivatives.

Control Using Linear Approximations: (S1, S2)

Consider a situation in which the values of the sensory function, $\emptyset(p_i)$ and its gradient,$\nabla\emptyset|_{p_i}$ are available continuously to robot i at its current position. In this case, the available information motivates a linear estimate of C_{v_i} over the known region v_i. We define the linear approximation to C_{v_i} calculated from an agent at position p_i as

$$\widehat{C_{v_i}} = \frac{1}{\widehat{M_{v_i}}} \int_{V_i^+} q\widehat{\emptyset}_i(q)dq, \qquad (10)$$

$$\widehat{M_{v_i}} = \int_{V_i^+} \widehat{\emptyset}_i(q)dq, \qquad (11)$$

$$V_i^+ = v_i \cap \{ q|^{\emptyset_i(q)} > 0\} \qquad and \qquad (12)$$

$$\widehat{\emptyset}_i(q) = \emptyset(p_i) + \nabla\emptyset^T|_{p_i}(q-p_i) \qquad (13)$$

The above formulation follows naturally from the definition of C_{v_i} and M_{v_i} in (5) and (4). The integrals are taken over V_i^+ to avoid calculating values of $M_{v_i} \leq 0$ and values of C_{v_i} outside of the region v_i. We can prove the stability of the proposed controller in the case of linear $\emptyset(q)$. In this case, the function $\emptyset(q)$ is fully parameterized by its value and gradient as measured at any point, therefore $\widehat{\emptyset}_i(q) = \emptyset(q)$. This special case becomes mathematically equivalent to that treated in [5]. Then the $\widehat{C_{v_i}} = C_{v_i}$ and from (9), we have:

$$\widehat{H_v} = -k \sum_{i=1}^{n} M_{v_i} e_i^T e_i$$

As we noted previously, $M_{v_i} > 0$, therefore $H_v \leq 0$ $\forall p_i$, and $H_v = 0$ if $p_i = C_{v_i}$ $\forall$i. Additionally, P_C is the largest invariant set since, from (7) and (8), $\dot{p}_i = 0$ $\forall$i. Then by **Lasalle's theorem,** $\lim_{t\to\infty} p_i = C_{v_i}$. $\forall v_i$Which implies P $\rightarrow P_C$

In the case of nonlinear $\emptyset(q)$, we observe that the control law with a linear estimation causes the robots to converge to Estimated centroid, C_{v_i}, of their Voronoi region. We call such configurations as near optimal. It is difficult to bound the error between the estimated centroid and the actual centroid in a general N-dimensional systems, where robots can move along an arbitrary curvilinear segment in three-space (e.g., a robot moving along a slope)

Efficient Computation of integrals: (S1)

We will use the convenient form off the centroid estimation above to derive an analytical expression which will make the control law feasible for robot platforms with limited computational resources. This eliminates the need to describe the Voronoi region and approximate the M_{v_i} and C_{v_i} integrals in a computationally- expensive numerical procedure.

Since the estimated $\widehat{\emptyset}_i$ is polynomial in q, we can use the results from (3) to find the integrals for M_{v_i} and C_{v_i}as polynomials in the vertices of the Voronoi region v_i. First, from (10), (11), and (13), we Can divide the integral expression, $\widehat{M_{v_i}}$ and $\widehat{C_{v_i}}$ into monomial components to get

$$\widehat{C_{v_i}} = \frac{1}{\widehat{M_{v_i}}}(\emptyset(p_i)\text{-}\nabla\emptyset^T|_{p_i}p_i)\int_{V_i^+} qdq + \frac{1}{\widehat{M_{v_i}}}\int_{V_i^+} q^T qdq\,\nabla\emptyset|_{p_i} \qquad (14)$$

Where $\widehat{M_{v_i}}$

$$\widehat{M_{v_i}} = (\emptyset(p_i)\text{-}\nabla\emptyset^T|_{p_i}p_i)\int_{V_i^+} dq + \nabla\emptyset^T|_{p_i}p_i\int_{V_i^+} dq \qquad (15)$$

These expressions can be simplified further by introducing the contacts $c_1 = (\emptyset(p_i)\text{-}\nabla\emptyset^T|_{p_i}p_i$, and $[c_1 \;\; c_2]^T = \nabla\emptyset|_{p_i}$ and by defining a general integral of a monomial over a polygon as

$$I_{V_i^+}^{\alpha\beta} = \iint_{V_i^+} x^\alpha y^\beta\, dxdy \qquad (16)$$

Where q =$[x \quad y]^T$. then we can write as

$$\widehat{C_{v_i}} = \frac{c_1}{\widehat{M_{v_i}}}\begin{bmatrix} I_{V_i^+}^{10} \\ I_{V_i^+}^{01} \end{bmatrix} + \frac{1}{\widehat{M_{v_i}}}\begin{bmatrix} I_{V_i^+}^{20} & I_{V_i^+}^{11} \\ I_{V_i^+}^{11} & I_{V_i^+}^{02} \end{bmatrix}\begin{bmatrix} c_2 \\ c_3 \end{bmatrix} \qquad (17)$$

Where

$$\widehat{M_{v_i}} = I_{V_i^+}^{00}c_1 + \begin{bmatrix} I_{V_i^+}^{10} & I_{V_i^+}^{01} \end{bmatrix}\begin{bmatrix} c_2 \\ c_3 \end{bmatrix}. \qquad (18)$$

The solution of the integral in (16) is given in analytical form by equation (4) from [3]. The cases specifically required for computation are simplified and enumerated below

$I_{V_i^+}^{00} = \frac{1}{2}\sum_{i=1}^{m}(y_{i+1} - y_i)(x_{i+1} + x_i)$,

$I_{V_i^+}^{01} = \frac{1}{6}\sum_{i=1}^{m}[(x_i - x_{i+1})(y_{i+1}^2 + y_i\ + 1y_i + y_i^2)\ +3(x_i + 1y_{i+1}^2 - x_i y_i^2)]$

$I_{V_i^+}^{10} = \frac{1}{6}\sum_{i=1}^{m}[(y_{i+1} - y_i)(x_{i+1}^2 + x_i\ + 1x_i + y_i^2)$,

$I_{V_i^+}^{11} = \frac{1}{24}\sum_{i=1}^{m}[(y_{i+1} - y_i)(2x_{i+1}x_i\,(y_{i+1} + y_i\ )+ x_{i+1}^2(3y_{i+1} + y_i) + x_i^2(y_{i+1} + 3y_i)]$,

$I_{V_i^+}^{02} = \frac{1}{12}\sum_{i=1}^{m}[(x_i - x_{i+1})(y_{i+1}^3 + y_{i+1}^2 y_i\ +\ y_i + 1y_i^2 + 1y_i^3)\ +4(x_i + 1y_{i+1}^3 - x_i y_i^3)]$,

$I_{V_i^+}^{00} = \frac{1}{2}\sum_{i=1}^{m}(y_{i+1} - y_i)(x_{i+1}^3 + x_{i+1}^2 x_i + x_i + 1x_i^2 + x_i^3)$,

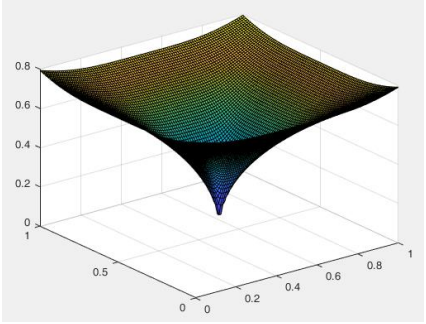
Where m is the number of vertices $[x_i\ y_i]^T$ of V_i^+ and where the index m+1 is interpreted as 1 (gets back in a cycle). The vertices must be ordered counterclockwise around the perimeter of V_i^+. The expression in (17) with the associated expressions for integral terms can be computed directly to give the actual value of $\widehat{C_{v_i}}$ and $\widehat{M_{v_i}}$

Validation In Simulations: (S1, S2)

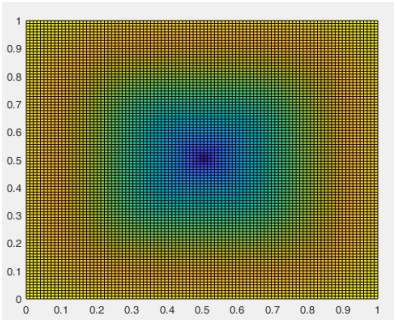
Simulations were carried out in MATLAB for various density functions. The number of robots and the number of time steps are varied to generate various results.

MATLAB Algorithm: (S1, S2)

- Defining variables: Number of Robots, Time-Steps, Speed of each robot, External Boundary, Initial Position of Robots (Randomly generated or uniformly placed points)
- Defining Video output File – All the frames will be merged to be shown in a span of 5 seconds (Time step can be 100Sec or 200Sec, the video length will be 10 seconds, can be changed at 'videoLength')
- Assigning our density Function $\emptyset(q)$ and $\nabla\emptyset(q)$. We are taking 3 different equations:
  - **Function-1:** $\emptyset(\boldsymbol{q}) = \sqrt[9]{(\boldsymbol{x}-\mathbf{0.5})^{\mathbf{4}}+(\boldsymbol{y}-\mathbf{0.5})^{\mathbf{4}}}$

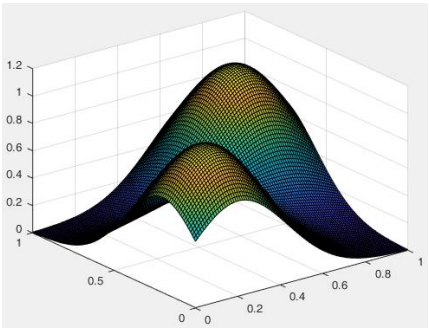


3D View of the Function

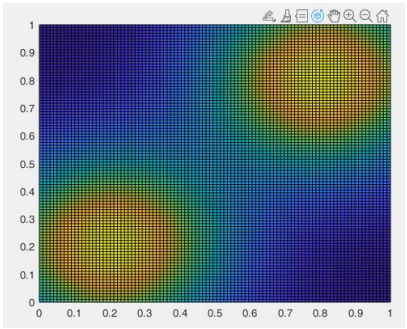


Top View of the Function

  - **Function-2:** $\emptyset(\boldsymbol{q}) = \boldsymbol{e}^{-\mathbf{9}((\boldsymbol{x}-\mathbf{0.2})^{\mathbf{2}}+(\boldsymbol{y}-\mathbf{0.,2})^{\mathbf{2}})} + \boldsymbol{e}^{-\mathbf{9}((\boldsymbol{x}-\mathbf{0.8})^{\mathbf{2}}+(\boldsymbol{y}-\mathbf{0.,8})^{\mathbf{2}})}$

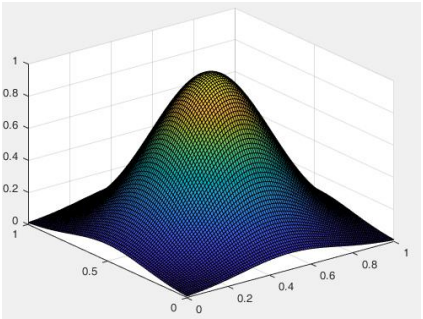


3D View of the Function

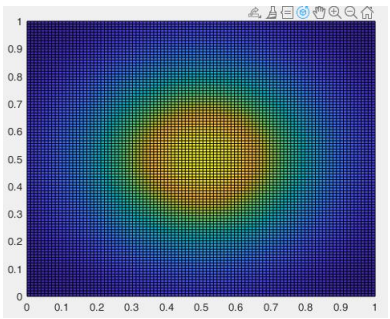


Top View of the Function

  - **Function-3:** $\emptyset(\boldsymbol{q}) = \boldsymbol{e}^{-\mathbf{9}((\boldsymbol{x}-\mathbf{0.5})^{\mathbf{2}}+(\boldsymbol{y}-\mathbf{0.,5})^{\mathbf{2}})}$



3D View of the Function



Top View of the Function

- Defining the variables (I_V) for calculating Weighted Centroid of Voronoi region C_{V_i}.
- Loop for time step:
    - Loop for each point:
        - Get all vertices of its Voronoi region
        - Evaluate constants c1, c2 and c3
        - Evaluating the values of $I_V^{00}, I_V^{01}, I_V^{10}, I_V^{11}, I_V^{02}, I_V^{20}$. They are calculated for vertices in anti-clockwise order.
        - Evaluating M_{V_i} and C_{V_i} of the Voronoi region
    - We have now obtained all the centroids of Voronoi regions
    - Loop for each point to move it towards the centroid. Each robot can move maximum of "speed" units per time step. (Variable can be adjusted at the beginning)
        - Taking values of distance moved by each robot in each time-step.
    - Generate a Voronoi plot frame of new coordinates for our video simulation.
- Generate plot of distance moved by any Robot Vs time.

MATLAB Code File:

Please find the file "Full Code" and run the file "Complete_Simulation.m". This will generate the initial and final position Voronoi graphs and create a video file showing the movement of robots. A graph of distance Vs time is also generated for the first three robots (can be changed for n robots).

Note: All the input variables are labelled and can be changed to generate a simulation accordingly. Another point to note is that all points are generated to be inside a square of opposite vertices (0,0) and (1,1).

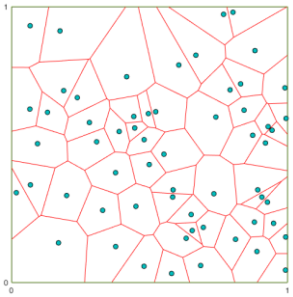
Video Files:

Please find the file "Simulation Results". It has 3 folders (Each folder corresponding to the density function), and each folder contains 2 videos (One for random initial points and one for uniformly distributed initial points).

Results:

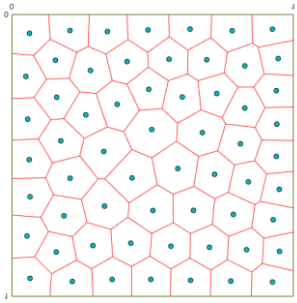
Function 1: $\emptyset(q) = \sqrt[9]{(x-0.5)^4 + (y-0.5)^4}$

Random Starting Points, 60 Robots, 100 Time Steps, Maximum Distance per time step = 0.1:



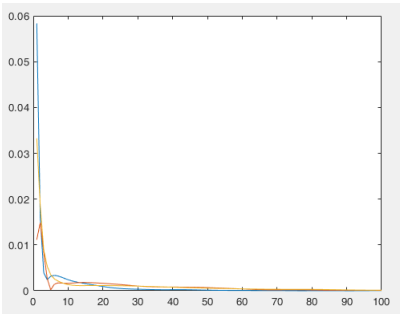
Figure(F1a)

Initial Coordinates



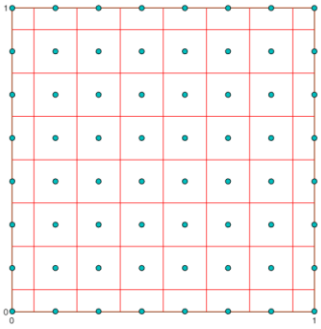
Figure(F1b)

Final Coordinate Points



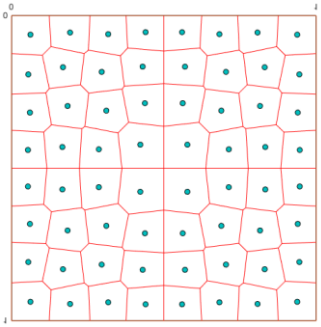
Figure(F1c): First 3 Distance Moved Vs Time

Evenly spaced Initial Points, 64 Robots, 100 Time Steps, Max Distance per time step = 0.1:



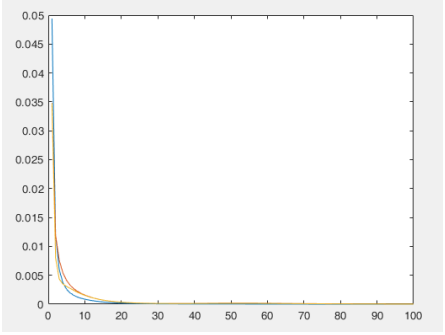
Figure(F1d)

Initial Coordinates



Figure(F1e)

Final Coordinate Points

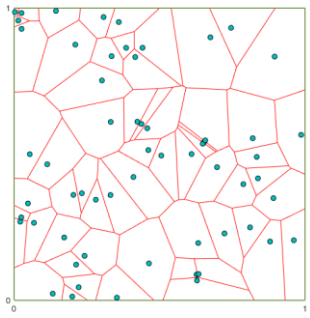


Figure(F1f): First 3 Distance Moved Vs Time

For both cases, we see that the robots move away from the centre and try to converge towards the border. Please look at the video files submitted for better understanding.

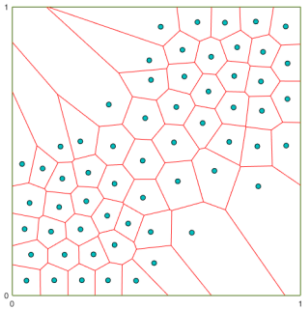
Function 2: $\emptyset(q) = e^{-9((x-0.2)^2+(y-0.,2)^2)} + e^{-9((x-0.8)^2+(y-0.,8)^2)}$

Random Starting Points, 60 Robots, 100 Time Steps, Maximum Distance per time step = 0.1:



Figure(F2a)

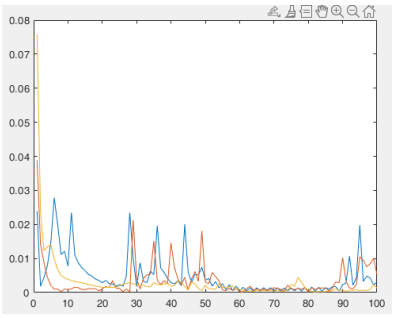
Initial Coordinates



Figure(F2b)

Points

Final Coordinate

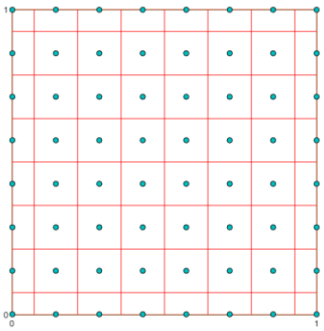


Figure(F2c): First 3

Time

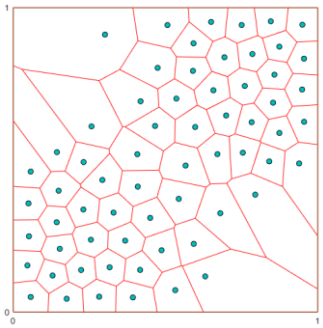
Distance Moved Vs

Evenly spaced Starting Points, 64 Robots, 100 Time Steps, Maximum Distance per time step = 0.1:



Figure(F2d)

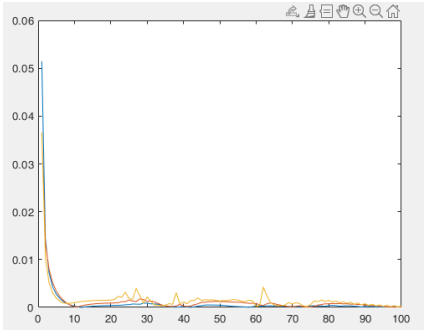
Initial Coordinates



Figure(F2e)

Points

Final Coordinate



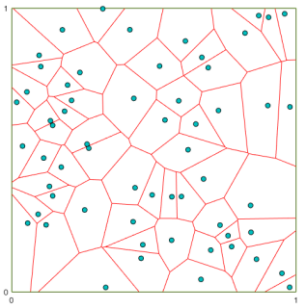
Figure(F2f): First 3

Distance Moved Vs Time

In both cases, the robots converge towards (0.2,0.2) and (0.8,0.8), the local maxima and the movement of each robot per time step converges to zero. Please look at the generated video files for better understanding.

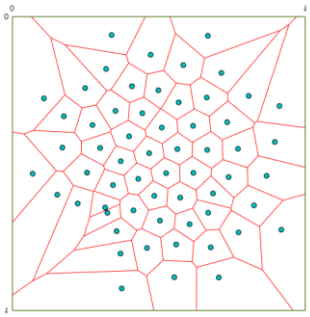
Function 3: $\emptyset(q) = e^{-9((x-0.5)^2+(y-0.,5)^2)}$

Random Starting Points, 60 Robots, 100 Time Steps, Maximum Distance per time step = 0.1:



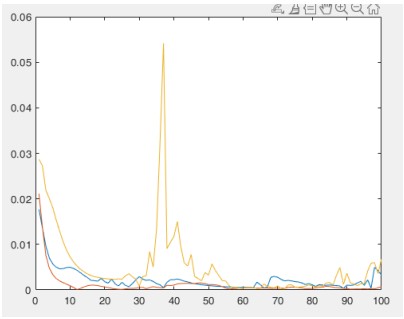
Figure(F3a)

Initial Coordinates



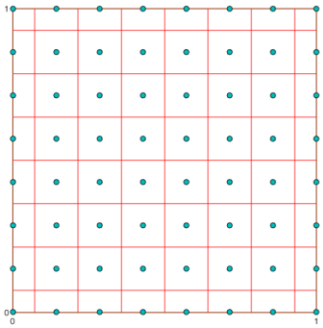
Figure(F3b)

Final Coordinate Points



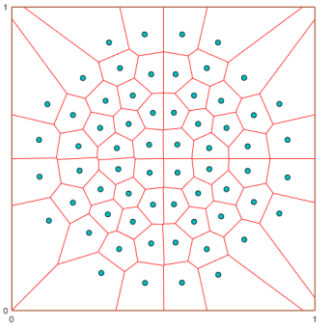
Figure(F3c): First 3 Time Steps Vs Distance Moved Vs

Evenly spaced Starting Points, 64 Robots, 50 Time Steps, Maximum Distance per time step = 0.1:



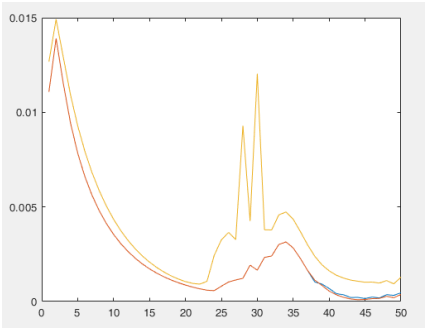
Figure(F3d)

Initial Coordinates



Figure(F3e)

Final Coordinate Points



Figure(F3f): First 3 Time Steps Vs Distance Moved Vs

In both cases, the robots converge towards the centre, which is the local maxima and the movement of each robot per time step converges to zero.

Note: The sudden spike in Distance Vs Time graph occurs when the robot is near the border. This is due to the centroid changing to a farther location with a small movement in Voronoi vertices. Another fact to consider is that we are not calculating the exact location of centroid in every step to reduce computation time and so, there will be some error.

Conclusion: In this paper, we were able to generate a decentralized method for controlling coverage in multi robot systems. We can also make the density function as a random function which changes with time and our system will be able to adapt to such changes. An analytical expression for centroid calculations is derived to make the algorithm computationally feasible. The expansions we can do to this model are numerous. One such expansion is implementing a sensory rating to each robot (higher the rating, more region can it cover). Studies on such models are still ongoing.

Acknowledgement:

This project is possible due to teachings from Associate Prof. Spring M. Berman in the course MAE 598 – Multi Robot Systems. We are grateful for your support.

References:

1) Event-based motion control for mobile sensor networks – By Zack Butler and Daniela Rus
2) Motion Coordination with Distributed Information – By Sonia Martinez, Jorge Cortes and Francesco Bullo
3) Distributed Coverage Control with Sensory Feedback for Networked Robots – By Mac Schwager, James Mclurkin and Daniela Rus
4) Efficient <Multi Robot Coverage of a known Environment – By Nare Karapetyan, Kelly Benson, Chris McKinney, Perouz Taslakain and Ioannis Rekleitis
5) Robust Adaptive Coverage Ccontrol for Robotic Sensor Networks – By Mac Schwager, Michael Vitus, Samantha Powers, Daniela Rus, and Claire Tomlin
6) C. Cattani and A. Paoluzzi. Boundary Integration over linear polyhedral. CAD
7) J. Cortes, S. Martinez, T. Karatas and F. Bullo – Coverage Control for mobile sensing networks.
8) Adapting to Sensing and Actuation Variations in Multi Robot Coverage by Alyssa Pierson, Lucas C, Luciano C A Pimenta and Mac Schwagger.
9) Marier J S, Rabbath, C. A and Lechevin – Optimizing location of sensors subject to health degradation
10) Michael, N. and Kumar, V. – Planning and control of ensembles of robots with non-holonomic constraints