Formation Control with Mobile Robots

Kadir Cimenci*

Middle East Technical University
kadircimenci@gmail.com

Abstract

Bitirme calismasindaki tum denklemler buradadÄśr

I. Introduction

$$A_i = \frac{A}{\zeta(c,N)(i+N)^c} \tag{1}$$

where $\zeta(c,N)$ is the Hurwitz zeta function defined by

$$\zeta(c,N) = \sum_{i=0}^{\infty} \left(\frac{1}{(i+N)^c} \right) \tag{2}$$

This known to converge for $c > 1$ and $N > 0$. In view of equation 2 one can write

$$\sum_{i=0}^{\infty} A_i = \sum_{i=0}^{\infty} \left(\frac{A}{\zeta(c,N)(i+N)^c} \right) \tag{3}$$

such that the sum of all areas A_i is the total area A to be filled, that is, if the algorithm does not halt then it is space-filling.

$$\hat{r_i} = \sqrt{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2} \quad (i = 1,2,...,n) \tag{4}$$

where i denotes the beacon number and n is the total number of beacons. We have n number of constraints in the solution of the localization problem. In our work, we have implemented a two dimensional localization solution with the assumption of each agent in the swarm have the same vertical position in Earth centered coordinate system. With this assumption, the problem for the localization process can be reduced down to a $A.\vec{x} = \vec{b}$ type linear system problem and the constraints will be circle functions rather than spherical ones, presented with

$$(x-x_i)^2 + (y-y_i)^2 = {r_i}^2 \tag{5}$$

Lets assume $\theta = (x,y)$ is representing the coordinates of an agent which is trying to localize itself, and $B1 = (x_1,y_1); B2 = (x_2,y_2); B3 = (x_3,y_3); ...; B_i = (x_i,y_i)$ are the agents with exactly known positions.

FOTOGRAF KONACAK

If any beacon is considered as the reference beacon and named with an index of r, the distance equations can be provided as following

*

The distance between the target agent and any beacon i

$$d_i(\theta) = \sqrt{((x - x_i)^2 + (y - y_i)^2)} \tag{6}$$

The distance between the referance beacon and the other beacons

$$d_i r(\theta) = \sqrt{((x_i - x_r)^2 + (y_i - y_r)^2)} \tag{7}$$

The distance between the target agent and the referance beacon

$$d_r(\theta) = \sqrt{((x - x_r)^2 + (y - y_r)^2)} \tag{8}$$

Adding and subtracting x_j, y_j and z_j in (6) gives

$$\begin{aligned} d_i^2(\theta) =& (x - x_r + x_r - x_i)^2 + (y - y_r + y_r - y_i)^2 \\ =& (x - x_r)^2 + 2(x_r - x_i)(x - x_r) + (x_r - x_i)^2 \\ &+ (y - y_r)^2 + 2(y_r - y_i)(y - y_r) + (y_r - y_i)^2 \end{aligned}$$

This equation yields to

$$2((x_i - x_r)(x - x_r) + (y_i - y_r)(y - y_r)) = d_r^2(\theta) + d_{ir}^2 - d_i^2(\theta)$$

this general statement is valid for each beacon with

$$\begin{aligned} &(x_2 - x_1)(x - x_1) + (y_2 - y_1)(y - y_1) = \frac{1}{2}[d_r^2(\theta) + d_{2r}^2 - d_2^2(\theta)] \\ &(x_3 - x_1)(x - x_1) + (y_3 - y_1)(y - y_1) = \frac{1}{2}[d_r^2(\theta) + d_{3r}^2 - d_3^2(\theta)] \\ &... \\ &(x_n - x_1)(x - x_1) + (y_n - y_1)(y - y_1) = \frac{1}{2}[d_r^2(\theta) + d_{nr}^2 - d_n^2(\theta)] \end{aligned}$$

if b_{ir} is defined for each beacon as follows:

$$b_{ir} := \frac{1}{2}[d_r^2(\theta) + d_{ir}^2 - d_i^2(\theta)] \tag{9}$$

then the linearized system equations can be represented with $A\vec{x} = \vec{b}$ type equation where;

$$A = \begin{bmatrix} x_2 - x_r & y_2 - y_r \\ x_3 - x_r & y_3 - y_r \\ ... & ... \\ x_n - x_r & y_n - y_r \end{bmatrix} \tag{10}$$

$$x = \begin{bmatrix} x - x_r \\ y - y_r \end{bmatrix} \tag{11}$$

$$b = \begin{bmatrix} b_{21} \\ b_{31} \\ ... \\ b_{n1} \end{bmatrix} \tag{12}$$

with the help of this mathematical manipulations, localization problem is reduced down to a $A\vec{x} = \vec{b}$ problem. There are some possible solutions to this type of equation regarding with the structure of matrix A and vector b.

SOLUTION TO Ax = b problem

In a localization problem handled in two dimensional world, the A matrix has $(n-1)$ rows and 2 columns, where $'n'$ is the number of neighbor beacons. It is obvious that there is no solution when the number of neighbors lower than 3 since the A matrix will have 1 or smaller number of lines. When the number of neighbor beacons are equal or greater than 3 we have three different solution types up to the structure of the linearized equations.

1) Unique solution If A matrix has the dimensions of $2x2$ and the rank of A matrix $'rank(A)'$ is equal to 2, then the solution of $\vec{x}$ is unique with

$$\hat{x} = A^{-1}\vec{b} \tag{13}$$

where $\hat{x}$ is the unique solution.

2) Minimum Norm solution with pseudo inverse

If A matrix has the dimensions of $(n-1)x2$ where $n > 3$,which means the number of neighbor beacons greater than 3, and if columns of A matrix form a linearly independent set (full column rank matrix) then the solution can be found with the projection of $\vec{b}$ over range space of A, $Proj_{R(A)}\vec{b}$ where

$$Proj_{R(A)}\vec{b} = A(A^TA)^{-1}A^T\vec{b} \tag{14}$$

$$A\vec{x} = Proj_{R(A)}\vec{b}$$
$$\vec{A}\hat{x} = A(A^TA)^{-1}A^T\vec{b}$$

with the help of the above equation

$$A(\hat{x} - (A^TA)^{-1}A^T\vec{b}) = 0 \tag{15}$$

then

$$\hat{x} = (A^TA)^{-1}A^T\vec{b} \tag{16}$$

since A matrix is full column rank matrix,

$$\mathcal{N}(\mathbf{A}) = \{0\} \quad and \quad \mathcal{N}(\mathbf{A})^{\perp} = \mathbb{R}^n$$

then

$$Proj_{\mathcal{N}(\mathbf{A})^{\perp}}\hat{x} = \hat{x} \tag{17}$$

this concludes that $\hat{x}$ is the unique minimum norm solution to the $A\hat{x} = \vec{b}$ problem

3) Minimum norm solution with nonlinear least squares method

If A matrix has the dimensions of $2x2$ or $(n-1)x2$ with $n > 3$ and if rank of A matrix is equal to 1, $rank(A) = 1$ then the solution to the $A\hat{x} = \vec{b}$ problem can be found iteratively with the help of nonlinear least squares method. Lets define the cost function to be minimized as the sum of the squares of the errors on the distances

$$F(\theta) = \sum_{i=1}^{n} \left(f_i^2(x,y) \right) \tag{18}$$

with

$$f_i(x,y) = \sqrt{(x-x_i)^2 + (y-y_i)^2} - r_i = f_i(\theta) \tag{19}$$

There are various algorithms to minimize the sum of the square errors in literature, Newton iteration is used to find the optimal solution in this work. Taking the partial derivatives of the cost function with respect to x and y gives

$$\frac{\partial F}{\partial \vec{x}} = 2 \sum_{i=1}^{n} f_i \frac{\partial f_i(\theta)}{\partial x}$$
$$\frac{\partial F}{\partial \vec{y}} = 2 \sum_{i=1}^{n} f_i \frac{\partial f_i(\theta)}{\partial y}$$

The partial derivative matrix of the cost function is composed as;

$$\bigtriangledown F(\theta) = 2 \begin{bmatrix} f_1 \frac{\partial f_1(\theta)}{\partial x} + f_2 \frac{\partial f_2(\theta)}{\partial x} + ... + f_n \frac{\partial f_n(\theta)}{\partial x} \\ f_1 \frac{\partial f_1(\theta)}{\partial y} + f_2 \frac{\partial f_2(\theta)}{\partial y} + ... + f_n \frac{\partial f_n(\theta)}{\partial y} \end{bmatrix} \tag{20}$$

Components of this partial derivative matrix converges to zero while the cost function iteratively optimized to a minimum point.

$$\bigtriangledown F(\theta) = 2J(\theta)^T f(\theta) = 0 \tag{21}$$

where

$$J(\theta) = \begin{bmatrix} \frac{\partial f_1(\theta)}{\partial x} & \frac{\partial f_1(\theta)}{\partial y} \\ \frac{\partial f_2(\theta)}{\partial x} & \frac{\partial f_2(\theta)}{\partial y} \\ ... & ... \\ \frac{\partial f_n(\theta)}{\partial x} & \frac{\partial f_n(\theta)}{\partial y} \end{bmatrix} \tag{22}$$

and

$$f(\theta) = \begin{bmatrix} f_1(\theta) \\ f_2(\theta) \\ ... \\ f_n(\theta) \end{bmatrix} \tag{23}$$

Using the vector $\vec{R}$

$$\vec{R} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{24}$$

To optimize the cost function, Newton iteration is implemented as follows;

$$\vec{R}_{\{k+1\}} = \vec{R}_{\{k\}} - (J_{\{k\}}^T J_{\{k\}})^{-1} J_{\{k\}}^T \vec{f}_{\{k\}} \tag{25}$$

where $\vec{R}_{\{k\}}$ denotes the approximate solution at k^{th} iteration. The explicit form of the equations can be derived by implementing our constraint functions to the generic statements, as follows;

$$J^T J = \begin{pmatrix} \sum_{i=1}^{n} \frac{(x-x_i)^2}{(f_i+r_i)^2} & \sum_{i=1}^{n} \frac{(x-x_i)(y-y_i)}{(f_i+r_i)^2} \\ \sum_{i=1}^{n} \frac{(x-x_i)(y-y_i)}{(f_i+r_i)^2} & \sum_{i=1}^{n} \frac{(y-y_i)^2}{(f_i+r_i)^2} \end{pmatrix} \tag{26}$$

and

$$J^T \vec{f} = \begin{pmatrix} \sum_{i=1}^{n} \frac{(x-x_i)f_i}{(f_i+r_i)} \\ \sum_{i=1}^{n} \frac{(y-y_i)f_i}{(f_i+r_i)} \end{pmatrix} \tag{27}$$

II. Route Table Determination and - DSDV Tablolar

It is obvious that each agent must have at least three neighbor agents to solve the $A\vec{x} = \vec{b}$ type problem in two dimensional domain and recalculate its position in the environment. Since the possibility of having a large error on position and velocity data for the agents which do not have an external position measurement sensors, it will be appropriate to get the agents into local trilateration process with the agents which have position sensors as much as possible. It is assumed that the positions of the beacon agents in the trilateration process are well-known with a little error boundary, ideally with no errors, so getting in the trilateration process with the agents which are already have errors on their position and velocity datas may increase the error on the calculated datas. On the other hand due to the restrictions & requirements defined in the Section 1-2 Objectives, it will not be possible to interact with the agents with agents with position sensors directly due to the small communication ranges of the agents and line of sight issues. In this case, it will be a good choice to handle this trilaterion process starting with the agents which are closer to the position agents in a increasing order of distance.

SUNUM1 deki yukaridan asagi biliginin dagitimi sekli buraya gelecek

As illustrated on the Figure-xx, suppose that the red agents which are closest ones to the position beacon in the swarm are the only ones which have the capability of interacting with the position beacon. Since the only source of true position measurement is the position beacon, this data must be distributed to these red agents first, then trilateration process must be propogated through the orange agents and then the yellow agents last, since they can only interact with an upper layer in the swarm due to the line of sight and communication range issues.

It is needed to have an algorithm to organize the order of the local trilateration process and the determines the beacon agents for each member of the swarm. Basically this algorithm will assign each agent a rank which represenets the number of sequence in the localization process, and will determine at least three local neighbors of each agent to get in trilateration. Agents which do no have at least three neighbors are assumend to be lost agents and the handling of these type of agents are illustrated in Section -xx.

I. Routing Algorithm- Bellman Ford

As mentioned in the Objectives section of the thesis work, agents are assumed to have a limited communication range and bandwith and the communication topology in the swarm is implemented with a wireless mesh network. In this type of network, each node have a relay in the

network and the data is transferred to the related destination with the help of route tables. This makes it possible to have the capability of transferring low bandwith data through the network with multiple hops. In this work, we implement this topology with a table driven routing scheme known as DSDV(Destination-Sequenced Distance Vector Routing Protocol) algorithm based on Bellman Ford algorithm. Bellman-Ford is and algorithm that computes the shortest path in a weighted graph and the correctness of the algorithm is proven. It is an algorithm based on relaxation, the correct distance to the vertices in the graph are updated iteratively from the initial estimations until converging to the optimal solution. This algorithm is slower than the Djikstra's algorithm which has similar functionalities but negative edge weights can be implemented in the related graph to report the negative cycles which means there is no cheapest path to the related destination vertex. On the other hand, it is possible to augment this algorithm with DSDV implementation to handle the routing loop problem when there is one or more vertices are no more exists in the network. The probability of the case with the non-existence of some vertices during the algorithm is processed can be very high since the agents in the swarm have low sensor capabilities and small range of communication and they have a great possibility to get lost in the environment. A simple demonstration of the Bellman-Ford algorithm with a simple network is illustrated in the Figure-xx

Powerpoint cizilen ornek network ve iterasyonlari eklenecek

The algorithm to calculate the shortest paths for node 'S' is done at the end of 7 iterations. At the beginning of the process, the weights for each edges are determined including the negative ones and each distance to the paths are filled with infinitive. Then the shortest paths to the each node in the given directed graph are determined iteratively with the help of the Bellman-Ford algorithm

I.1 Usage on Bellman Ford algorithm and DSDV

Bellman Ford algorithm have a drawback related with the routing loop problem which occurs in an event of one or more nodes in the graph are lost during the process. Figure -xx illustrates a simple routing loop problem.

POwerpoint deki sunum-1 den alinan cizim eklenenecek

Suppose that the node D have lost its contact with the network due to some malfunction or being lost by getting outside of the communication range to the closest neighbor of itself. Before this event,node C have a unit distance to the node D and consequently node B have a 2 unit distance to node D , node A have a 3 unit distance to node D. In case of a failure on node D, on the next iteration C will update its route table with the 3 unit distance to node D by taking reference the node B. Then node B will update its route table with the shortest distance of 4 units to the node D by referencing the node C and this process will diverge to infinity on the shortest paths with the increasing number of iterations. To provide a solution for this type of problems, DSDV algorihm has implement the sequence numbers and counts for hops into the route tables of the nodes. A simple route table for a vertex in a network is given in Figure -xx

Metric ve dest.seq iceren resim buraay eklenecek

In the DSDV algorithm, each node have a sequence number and counts for hops (metric) for each route in its route table and periodically transmits the updates including its own sequence number and routing tables updates. In the network, when two routes to the same destination received from two different neighbors the nodes will observe the following rules;

- Choose the one with the larger destination sequence number - If the sequence numbers are equal, then choose the route with minimum number of hops and update the route table.

DSDV Link addition

Sunumdan ilgili sekli koyalim

When a new node A joins the network, it transmits of its own route table including the destination to itself $< A, A, 0, 101 >$. Then the following procedure will be handled during iterations -Node B receives the the transmission of A and inserts a new line into its route table with $< A, A, 1, 101 >$ and propogates this new node to its neighbors -Node C and Node D receives this transmission and inserts the new route to their route tables with $< A, B, 2, 101 >$

DSDV link breaks

Sunumdan ilgili sekli koyalim

When the link between B and D breaks, node B gets no trasnmission from the D and notices the link breaks, then the following procedure will be handled, - Node B update the hop count for node D and E to the infinity and increments the sequence numbers to these routes - Node B propogates the updates to its neighbors and node A and node C updates the lines of the routes to the D and E, since the message from B includes higher sequence numbers for those routes.

DSDV is implementing an algorithm to find the shortest paths between the internal nodes of a given directed graph. The costs for each shortest paths are calculated with the help of the weight of edges in the graph. Since our aim is to find the closest position beacon which has the minimum number of hops, the weight for each edge in the graph must be represented with the same unit size and the directions of the edges are negligible.

II. clusters

Since there are limited number of position beacons in a swarm, it will be appropriate to cluster the agents around these position agents to minimize the problem to the subproblems in which every one of them there are only one position beacon and the agents which are assigned to that cluster. The error on the trilateration process is expected to be increasing at the lower layers of the process illustrated in Figure -xx because of the cumulative effects of errors added to the position and velocity data of the agents in each layer. Thus, the policy for the assignment of the agents to the clusters must be the number of hops to the routes of position beacons rather than the physical distances. Since DSDV algorithm has a structure storing the number of counts to each route in the tables, this information can be used to determine each agents' clusters in the swarm.

clusterlarla ilgili bir resim koyalim

As illustrated in the Figure -xx, all agents have assigned themselves to the clusters around position beacons, in which they have minimum number of hops in the route to the related position agent. With this approach, the generic algorithm which will be executed with the period of localization process for each agent must be implemented as follows:

1) Update route table including the routes to the position agents in the swarm with DSDV algorithm 2) Check for the routes to the position agents and join the cluster in which minimum number of hops required to that destination. 3) Wait for the localization sequence in which the agents are entered the process with the increasing number of hops, e.g. the agent which have a single hop to the position agents are processed first, and then the other ones are processed consecutively as illustrated in Figure -xx 4) If the localization sequence is valid(siranin gelmesi denmeli) for this agent, enter the trilateration process with the agents from upper layer, which are the next hops in the route table. 5) Call the update procedure of the observer system of which details are presented in section -xx

III. Handling Lost Agents

The minimum number of neighbors required for the trilateration process is three for a two dimensional localization problem as illustrated in section -xx(trilateration bolumu) . Since the

agents are assumed to have a narrow communication range, it is possible to not to find three neighbors for any agent at an instant time. At this case, it will be impossible to relocate these agents with trilateration and the position&velocity data will drift from the real values with the increasing time passed without trilaterions. To avoid these kind of problems, the concept of 'lost' agents and the procedures for these type of agents are described as follows:

* An agent gets into 'Lost' mode, if it doesn't find three neighbors at an instant time * If an agent is in 'Lost' mode and missed the localization process for three times, it will get into 'Return to Home' mode * If an agent is in 'Return to Home' mode, it will directly try to reach to the center of the desired formation shape.

The idea behind the 'Return to Home' mode is basically to increase the possibility of the lost agent to get in touch with the rest of the swarm with directing it to the center of the swarm. A simple demonstration of this procedure is illustrated in Figure-xx

Return to home sekli sunumdan konacak

The lost agent aims to reach to the center of the formation and due to the errors on its position&velocity data, it is expected to arrive to the red point illustrated in the figure. With this maneuver, the lost agent still have a chance to meet some other agents in the swarm even if it directs itself to an incorrect goal state.

IV. State Estimation Procedure

In local positioning subsystem, agents are expected to execute a state estimator algorithm in which they propogate their state vector composed of translational position and velocities with the help of inertial measurements. As discussed in Section 3.1 they will update and correct their positions with the measurements provided by the trilateration process exeeceuted with the localization timer period of 5 seconds. A Kalman estimator algorithm which uses the trilateration outputs as external measurents and the sensor measurement as inputs is designed to fusion the sensor measurements with the trilateration calculations. The model for this observer system is defined as follows:

The state vector for each agent is defined as:

$$x_k = \begin{bmatrix} X_k \\ \dot{X}_k \end{bmatrix} \tag{28}$$

where X_k is the position and $\dot{X}_k$ is the velocity of the agents in x coordinates in two dimensional environment. All of the following procedures will be handled exactly the same for the state vector in y coordinates. The linear model to propogate states will be:

$$x_{k+1} = F_k x_k + B_k u_k + w_k \tag{29}$$

where w_k is the process noise and

$$F = \begin{bmatrix} 1 & d_t \\ 0 & 1 \end{bmatrix} \tag{30}$$

$$B = \begin{bmatrix} \frac{{d_t}^2}{2} \\ d_t \end{bmatrix} \tag{31}$$

where d_t is the propogation period and u_k is the translational acceleration measured by inertial sensors in the x coordinate of the system. The observation which will be calculated with the trilateration process :

$$z_k = H_k x_k + v_k \tag{32}$$

where v_k is the measurement noise and since the trilateration process will provide new position informations of the agents:

$$H_k = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{33}$$

The noise models for the process and the measurement are modelled with:

$$w_k = \mathcal{N}(\mathbf{0}, \mathbf{Q_k}) \tag{34}$$

$$v_k = \mathcal{N}(\mathbf{0}, \mathbf{R_k}) \tag{35}$$

where w_k is the process noise with zero mean multivariate normal distribution with covariance of Q_k and v_k is the measurement noise with zero mean Gaussian distribution with a covariance of R_k

The filter has two main subsections named predict and update phases. The update phase of the filter is executed after each trilateration process with a period of 5 seconds. The filter equations are as follows:

Propogation phase:

$$\hat{x}_{k,k-1} = F_k \hat{x}_{k-1,k-1} + B_k u_k \tag{36}$$

$$P_{k,k-1} = F_k P_{k-1,k-1} F_k^T + Q_k \tag{37}$$

Update Phase:

$$\tilde{y}_k = z_k - H_k \hat{x}_{k,k-1} \tag{38}$$

$$S_k = H_k P_{k,k-1} H_k^T + R_k \tag{39}$$

$$K_k = P_{k,k-1} H_k^T S_k^{-1} \tag{40}$$

$$\hat{x}_{k,k} = \hat{x}_{k,k-1} + K_k \tilde{y}_k \tag{41}$$

$$P_{k,k} = (I - K_k H_k) P_{k,k-1} \tag{42}$$

where Q_k is the process covariance matrix and R_k is the measurement covariance chosen as

$$Q_k = \begin{bmatrix} Max.AccelerationError * \frac{d_t^2}{2} & 0 \\ 0 & Max.AccelerationError * d_t \end{bmatrix} \tag{43}$$

$$R_k = Max.PositionErroronTrilaterationProcess \tag{44}$$

in the above equations K_k represents the Kalman gain matrix and S_k is the residual covariance of the system at time k. $\hat{x}_{k,k}$ is the posteriori state estimate updated with measurements at time k ; $\hat{x}_{k,k-1}$ is the priori estimate of the state vector predicted with inputs at time k; $P_{k,k}$ is the posteriori error covariance matrix updated with measurements at time k; $P_{k,k-1}$ is the priori estimate covariance predicted with the inputs at time k

III. FORMATION CONTROL

The details of the methodology for dynamical formation control with heterogenous mobile robots is presented in this chapter. Basically three different approaches as artificial forces method, bubble packing method and randomized fractals method are implemented. It is possible to classify these methods in two sub categories. Potential field based approaches implements artificial forces acting on agents to get inside and cover the desired formation shape homogenously by avoiding collisions between the agents. The resultant positions of the agents in the formation shape is not certain, it dynamically changes with the instantaneous positions and interactions of the agents with each

other and environment. The other two methods , shape partitioning based approaches, share a common structural basis. In these approaches the complex formation shape is partitioned into goal states to cover the shape homogenously with the mobile robots. The assignment of the agents to these goal states is handled with special algorithms to optimize the overall energy consumpytion of the swarm. The difference between these two methods is the partitioning approach of the complex shape.

Method lari gosteren sekli buraya ekleyelim

I. artificial forces method

Artificial forces method implements some potential fields over each agent arised from the interactions between agents, formation shape and environment etc. The positions of the agents at the formation shape are determined randomly with a local equilibrium point of the swarm in which every agent is at balance under the total force acting from the environment. There are basically three different kinds of artificial forces named intermember forces representing the forces created by the other agents in the swarm to achieve collision avoidance, the attractive forces representing the forces created by the desired formation shape to attract the agent into the shape and repulsive forces created by the formation shape to keep agent inside the shape. It is possible to augment these type of forces for specific tasks and objectives, e.g. obstacle forces created by the obstacles in the environment can be implemented to achieve obstacle avoidance. Since the method to calculate the artificial forces involves contour integrals, it will be useful to give mathematical definition of contour integrals;

Consider a curve C which is a set of points $z = (x, y)$ in the complex plane defined by

$$x = x(t),\ y = y(t),\ a \leq t \leq b \tag{45}$$

where $x(t)$ and $y(t)$ are continuous functions of the real parametere t. It is possible to write

$$z(t) = x(t) + iy(t),\ a \leq t \leq b \tag{46}$$

This curve is called smooth if $z(t)$ has continuous derivative $z'(t) \neq 0$ for all points along the curve, and it is called simple if it does not cross itself as mentioned;

$$z(t_1) \neq z(t_2)\ whenever \quad t_1 \neq t_2 \tag{47}$$

On the other hand if $z(a) = z(b)$ is the only intersection point, the curve is said to be simple closed curve. Regarding with these given definitions, an example for a simple smooth closed curve is illustrated in Figure -xx

Sunumdan simple closed curve buraya konacak

Let $f(z)$ is a complex function in a domain D in the complex pane and let C be simple closed contour contained in D with initial point z_0 and terminal point z. It is possible to take the integral of $f(z)$ along the contour C

$$\oint_C f(z)dz = \int_a^b f(z(t))\frac{dz(t)}{dt}dt \tag{48}$$

where

$$\frac{dz(t)}{dt} = \frac{dx(t)}{dt} + i\frac{dy(t)}{dt},\ a \leq t \leq b \tag{49}$$

To simplify this equation, one can wrtie $f(z) = u(x, y) + iv(x, y)$ and $dz = dx + idy$ into the statements,

$$\begin{aligned}\oint_C f(z)dz &= \oint_C udx - vdy + i\oint_C udy + vdx \\ &= \int_a^b \left[u(x(t),y(t))\frac{dx(t)}{dt} - v(x(t),y(t))\frac{dy(t)}{dt}\right]dt \\ &\quad + i\int_a^b \left[u(x(t),y(t))\frac{dy(t)}{dt} + v(x(t),y(t))\frac{dx(t)}{dt}\right]dt\end{aligned}$$

II. Artificial Forces and Utitilty Functions

As it is mentioned in the Section-xx Objectives part, the formation shapes will be complex contours which cannot be identified analytically, but the definition for the artificial forces and utility functions are expressed in continuous contour integrals which requires the analytical expression of the curve on which the integral will be taken. To enable these type of calculations it is required to provide these statements in discrete domain to achieve calculations with complex closed curves.

1- Cauchy Winding Number Cauchy winding number of a curve in the plane around a given point is the number of times that curve travels counterclockwise around the point. This number is used to switch on/off some of the artificial forces while the agent is inside or outside of the formation shape. Suppose C is the complex closed curve which is a set of points $z = (x, y)$ in the complex plane and z_i is a point to check whether it is inside of the curve, then the Caucy winding number is :

$$n(C, z_i) = \frac{1}{2\pi i}\oint_C \frac{dz}{z - z_i} \tag{50}$$

The winding number for agent i in the swarm,

$$n(C, z_i) = \begin{cases} 1 & \text{when member i is inside C} \\ 0 & \text{when member i is outside C} \end{cases} \tag{51}$$

To implement this statement in discrete domain

$$n(C, z_i) = \frac{1}{2\pi i}\oint_C f(z)dz \tag{52}$$

where

$$f(z) = \frac{1}{z - z_i} \tag{53}$$

Function of $f(z)$ can be partitioned into real and complex parts as:

$$u(x, y) = real(f(z)) \;\; and \;\; v(x, y) = imag(f(z)) \tag{54}$$

partitioning as it mentioned in equation -xx

$$\oint_C f(z)dz = \oint_C udx - vdy + i\oint_C udy + vdx \tag{55}$$

then

$$n(C, z_i) = \frac{1}{2\pi i}\left[\int_a^b \left(u\frac{dx}{dt} - v\frac{dy}{dt}\right)dt + i\int_a^b \left(u\frac{dy}{dt} + v\frac{dx}{dt}\right)dt\right] \tag{56}$$

This contour integral representation of this equation is

$$n(C,z_i)=\frac{1}{2\pi i}\left[\sum_{k=1}^{K}\left(u(x_{k+1}-x_k)-v(y_{k+1}-yx_k)\right)+i\sum_{k=1}^{K}\left(u(y_{k+1}-y_k)+v(x_{k+1}-x_k)\right)dt\right] \tag{57}$$

where

$$z_k-z_{k-1}=z_{k+1}-z_k,\ \ \forall k;\ \ when\ \ K\to\infty \tag{58}$$

The assumption of $K\to\infty$ makes it possible to calculate the integral of Cauchy winding number with a small error with large number of K which can be achieved by partitioning the desired formation shape into small pieces with equal distances. This approach is used to provide representations of the contour integrals in discrete domain.

Burada iceride dÄśsarda noktalarla doldurdughumuz seklÄś verelim

2- Length of a formation shape

The length of a formation shape can be calculated with the equation of;

$$l(C)=\oint_C \|dz\| \tag{59}$$

the expression for this contour integral with points of $z_k=(x_k,y_k)$ in the complex plane

$$l(C)=\sum_{k=1}^{K}\sqrt{(x_{k+1}-x_k)^2+(y_{k+1}-y_k)^2} \tag{60}$$

where

$$z_k-z_{k-1}=z_{k+1}-z_k,\ \ \forall k;\ \ when\ \ K\to\infty \tag{61}$$

3-Center of a Formation Shape

The center of a formation shape can be calculated with the equation of;

$$z_c=\frac{\oint_C z\,\|dz\|}{l(C)} \tag{62}$$

the expressiion for this equationl with points of $z_k=(x_k,y_k)$ in the complex plane

$$z_{cx}=\frac{\sum_{k=1}^{K}x(k)}{K}$$
$$z_{cy}=\frac{\sum_{k=1}^{K}y(k)}{K}$$

where z_{cx} and z_{cy} are the x and y coordinates of the center of formation shape respectively.

4- Area of a Formation Shape

Green's theorem can be used to calculate of the area of a closed curve. According to this theorem the area of D given by the double integral

$$A=\int\int_D dA \tag{63}$$

can be calculated with the line integral of

$$A=\oint_D Fds=\frac{1}{2}\oint_D xdy-ydx \tag{64}$$

where

$$F(x,y)=(-y/2,x/2) \tag{65}$$

This contour integral can be reduced down to

$$\begin{aligned} Area &= \frac{1}{2}\oint_C xdy - \frac{1}{2}\oint_C ydx \\ &= \frac{1}{2}\int_{t=a}^{b} x(t)\frac{dy(t)}{dt}dt - \frac{1}{2}\int_{t=a}^{b} y(t)\frac{dx(t)}{dt}dt \end{aligned}$$

the expressiion for this equationl with points of $z_k = (x_k, y_k)$ in the complex plane

$$Area = \frac{1}{2}\sum_{k=1}^{K} x_k(y_{k+1} - y_k) - \frac{1}{2}\sum_{k=1}^{K} y_k(x_{k+1} - x_k) \tag{66}$$

where

$$z_k - z_{k-1} = z_{k+1} - z_k, \ \ \forall k; \ \ when \ \ K \to \infty \tag{67}$$

I- ARTIFICIAL FORCES Artificial forces are defined to gather the agents in the swarm inside a formation shape and make them distributed homogenously inside the shape. It is possible to define some additional artificial forces to implement features like obstacle&collision avoidance or smooth transitions between the boundaries of the formation shape. Attractive forces, repulsive forces, intermember forces, obstacle forces and transition forces are implemented to generate individual control laws for all agents in the swarm. Suppose the state of a member i is described by

$$X_i = \begin{bmatrix} z_i \\ \dot{z}_i \end{bmatrix} \tag{68}$$

where $z_i \in C$, represents the position of the i^{th} member of the swarm and $\alpha = \begin{bmatrix} 1 & 0 \end{bmatrix}$ and $\beta = \begin{bmatrix} 0 & 1 \end{bmatrix}$. The state of the whole swarm $x = \begin{bmatrix} X_1 & X_2 & ...X_n \end{bmatrix}$ is determined by the linear equations of [pubudu referans verelim]

$$\dot{x} = Ax + Bu \tag{69}$$

where

$$\begin{aligned} A &= diag\left(\hat{A}\right)_{nxn} \\ B &= \frac{1}{m} diag\left(\hat{B}\right)_{nxn} \end{aligned}$$

with

$$\hat{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \ \ \hat{B} = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{70}$$

The vector for individual control laws of the swarm

$$u = \begin{bmatrix} u_1 & u_2 & ... & u_n \end{bmatrix} \tag{71}$$

where

$$u_i = F_{i,a} + F_{i,r} + F_{i,m} + F_{i,t} \tag{72}$$

It is necessary to define the concept of "coveage circle" for the agents which will be used in the artificial forces calculations. Coverage circle of agent i, C_i is defined as the circle with minimum radius which can cover the whole agent's collision surface. The radius of this circle is given with d_c. Some of the examples of coverage circles for different types of mobile robots are illustrated in Figure -xx below

Farkli sekillerde agentlar ve bunlarin ertrafindaki daireleri gosteren coverage circle resimleri koyalim

The components of these control forces are described in details at the following section. 1- Attractive Forces Attractive forces are the artificial force components generated by the formation shape to attract the agent towards the center of the formation .They are active when the agents are outside of the shape.

Attractive force lari gosteren bir sekil koyalim, agentlara shape disarisindayken etki eden kuvvet vektorleri

The equations for the attractive forces are defined as follows:

$$F_{i,a} := \frac{k_a(1 - n(C, \alpha X_i))}{l(C)} \oint_C (z - \alpha X_i) \, \|dz\| \tag{73}$$

where k_a is the variable gain for the attractive forces. The representation of the attractive forces on agent i on $z_i = (x_i, y_i)$ with the points of $z_k = (x_k, y_k)$ of formation shape in the complex plane:

$$F_{iax} = \frac{k_a(1 - n(C, \alpha X_i))}{l(C)} \sum_{k=1}^{K} (x_k - x_i)$$
$$F_{iay} = \frac{k_a(1 - n(C, \alpha X_i))}{l(C)} \sum_{k=1}^{K} (y_k - y_i)$$

where

$$z_k - z_{k-1} = z_{k+1} - z_k, \;\; \forall k; \;\; when \;\; K \to \infty \tag{74}$$

and F_{iax}, F_{iay} are the attractive force components in x, y coordinates respectively.

2- Repulsive Forces Repulsive forces are the artificial force components generated by the formation shape to keep the agents inside the shape. They are active when the agents are inside the shape.

Repulsive force lari gosteren bir sekil koyalim, agentlara shape icindeyken etki eden kuvvet vektorleri The equations for the attractive forces are defined as follows:

$$F_{i,r} := k_r n(C, \alpha X_i) \oint_C \left[\frac{\alpha X_i - z}{\|\alpha X_i - z\|^3} \right] \|dz\| \tag{75}$$

where k_r is the variable gain for the repulsive forces. The representation of the repulsive forces on agent i on $z_i = (x_i, y_i)$ with the points of $z_k = (x_k, y_k)$ of formation shape in the complex plane:

$$F_{irx} = k_r n(C, \alpha X_i) \sum_{k=1}^{K} \frac{x_i - x_k}{\|\alpha X_i - z_k\|^3}$$
$$F_{iry} = k_r n(C, \alpha X_i) \sum_{k=1}^{K} \frac{y_i - y_k}{\|\alpha X_i - z_k\|^3}$$

where

$$z_k - z_{k-1} = z_{k+1} - z_k, \;\; \forall k; \;\; when \;\; K \to \infty \tag{76}$$

and F_{irx}, F_{iry} are the repulsive force components in x, y coordinates respectively.

3- Inter-member repulsion forces Intermember forces are the artificial force components generated by the agents in the swarm to avoid collisions between themselves.

Intermember kuvvetleri gosteren bir sekil ve mesafeye bagli olarak kuvvetlerin arttigini gosteren bubble packing deki sekil konsun

The equations for the intermember forces on the agent i on $z_i = (x_i, y_i)$ in the complex plane ,

$$F_{i,m} = k_m \sum_{j=1, j\neq i}^{N} \frac{\alpha X_i - \alpha X_j}{(\|\alpha X_i - \alpha X_j\|)} \frac{1}{(\|\alpha X_i - \alpha X_j\| - d_o)^2} \tag{77}$$

where d_o is the total distance minimum distance between the agents which can be calculated with

$$d_o = d_i + d_j \tag{78}$$

The force components in x,y coordinates respectively,

$$F_{imx} = k_m \sum_{j=1, j\neq i}^{N} \frac{x_i - x_j}{\|\alpha X_i - \alpha X_j\|} \frac{1}{(\|\alpha X_i - \alpha X_j\| - d_o)^2}$$
$$F_{imy} = k_m \sum_{j=1, j\neq i}^{N} \frac{y_i - y_j}{\|\alpha X_i - \alpha X_j\|} \frac{1}{(\|\alpha X_i - \alpha X_j\| - d_o)^2}$$

where k_m is the variable gain for the intermember forces.

4- Transition Forces

Transition forces are the artificial force component to force the agent inside the formation shape when they are close to the boundaries. Since the attractive forces have a decreasing nature while the agent getting closer to the formation shape, it is needed to add this type of forcing function to ensure the agents to get inside the shape. Transition forces are active outside of the desired formation shape.

Burada attractive force larin sekil boundary sine yaklastikca kuculdupu transition larin basladigini gosteren bir sekil konulsun

The equation for the transition forces are defined as follows:

$$F_{i,t} = k_t(1 - n(C, \alpha X_i) \oint_C \frac{z - \alpha X_i}{\|z - \alpha X_i\|} \|dz\| \tag{79}$$

where k_t is the variable gain for the transition forces. The representation of the transition forces on agent i on $z_i = (x_i, y_i)$ with the points of $z_k = (x_k, y_k)$ of formation shape in the complex plane:

$$F_{itx} = k_t(1 - n(C, \alpha X_i) \sum_{k=1,}^{K} \frac{x_k - x_i}{\|\alpha X_i - z_k\|^3}$$
$$F_{ity} = k_t(1 - n(C, \alpha X_i) \sum_{k=1,}^{K} \frac{y_k - y_i}{\|\alpha X_i - z_k\|^3}$$

where F_{itx}, F_{ity} are the transition force components in x, y coordinates respectively.

5- Obstacle forces

Obstacle forces are the artificial force components generated by the obstacle in the environment to achieve obstacle avoidance of the agents during formation control. The equation for the obstacle forces are defined as follows:

$$F_{i,o} := k_o \oint_O \left[\frac{\alpha X_i - z_o}{(\|\alpha X_i - z_o\| - d_{ci})^4} \right] \|dz_o\| \tag{80}$$

where k_o is the variable gain for the transition forces and d_{ci} is the radius of coverage circle of agent i . This contour integral is taken on the curve of the obstacle with points of $z_o = (x_o, y_o)$ in the complex plane.

The representation of the obstacle forces on agent i on $z_i = (x_i, y_i)$ with the points of $z_{ok} = (x_{ok}, y_{ok})$ of formation shape in the complex plane:

$$F_{iox} = k_o \sum_{k=1}^{K} \frac{x_i - x_{ok}}{(\|\alpha X_i - z_{ok}\| - d_{ci})^4}$$
$$F_{ioy} = k_o \sum_{k=1}^{K} \frac{y_i - y_{ok}}{(\|\alpha X_i - z_{ok}\| - d_{ci})^4}$$

where

$$z_{ok} - z_{ok-1} = z_{ok+1} - z_{ok}, \ \ \forall k; \ \ when \ \ K \to \infty \tag{81}$$

III. Buffer Zone Implementation

The attractive and transition forces are defined to be active when the agents are outside of the shape, the resistive forces are active when agents are inside the shape. Due to this type of sharp transitions on the total artificial force acting on an agent which is crossing the boundary of the formation shape, it is possible to have a chattering effect. To avoid this chattering effect and provide a smooth transition of the agent to pass the boundary of formation shape, a buffer zone around the boundaries are implemented. The main approach during the transition of these boundaries is to dynamically change the variable gains of these artificial forces to linearly change between zero and nominal values at the boundary conditions.

Linear degisen gainlerle iliskili bi sekil koyalim. hem att, trans. repusive icin

IV. XSwarm Theory

Samitha and Pubudu [xx] have provided a definition of "X-Swarm" and give a theorem which describes the motion of an agent towards s the formation shape based on the X -Swarm definition. Let S be a swarm with N agents. This swarm is defined as X-Swarm if there exists some positive constants δ and γ that satisfy the following conditions simultaneously for all $i, j \in S$ and $i \neq j$

- $d_{ij} \geq \gamma + \delta$
- $\left\| \frac{z_i - z_{cm}^i}{z_i - z_{cm}} \right\| < \left(1 + \frac{\delta}{\gamma}\right)^3$

where

$$d_{ij} = \|z_i - z_j\| \, and z_{cm}^i = \frac{\sum_{j=1;j\neq i}^{N} z_j}{N-1} \tag{82}$$

The term of z_{cm}^i is the center of mass of the swarm without the i^{th} member.

Lemma 1: The magnitude of artificial inter-member force acting on an agent of X-Swarm is less than

$$\frac{k_m(N-1)}{\gamma^3}\left\|z_i - z_{cm}\right\| \tag{83}$$

Proof:

$$F_{i,m} = k_m \sum_{j=1,j\ neqi}^{N} \frac{z_i - z_j}{d_{ij}^3} \tag{84}$$

Using the first condition of the X-Swarm , we have

$$\|F_{i,m}\| < \frac{k(N-1)}{(\gamma+\delta)^3}\left\|z_i - z_{cm}^i\right\| \tag{85}$$

Then using the condition 2 of the X-swarm definition, the following expression can be derived

$$\|F_{i,m}\| < \frac{k(N-1)}{\gamma^3}\left\|z_i - z_{cm}^i\right\| \tag{86}$$

This result provides an upper bound to the magnitude of the inter-member repulsion force in a X-Swarm.

Theorem: Conside the agent i outside of a desired formation shape in a X-swarm, if $\frac{k_a}{k_m} > \frac{N-1}{\gamma^3 l(C)}$ then the motion of member i is towards to the center of the swarm. Proof: Choosing a Lyapunov function candidate for the agent i as?

$$V_i = \frac{1}{2}\dot{v}_i\dot{v}_i^T + \frac{1}{2}v_i v_i^T\left(k_a l(C) - \frac{k_m(N-1)}{\gamma^3}\right) \tag{87}$$

It is possible to show that $\dot{V}_i$ is bounded by taking the derivatives,

$$\dot{V}_i \leq -k_f\left\|\dot{v}^2\right\| + \begin{bmatrix}\left\|k_m\sum_{j=1,j\neq i}^{N}\frac{z_i - z_j}{\left\|z_i - z_j\right\|^3}\right\| \\ -\frac{k_m(N-1)}{\gamma^3}\|v\|\end{bmatrix}\|\dot{v}\| \tag{88}$$

Since agent i is considered a member of a X-swarm, from Lemma -1

$$\left\|\sum_{j=1,j\neq i}^{N}\frac{z_i - z_j}{\left\|z_i - z_j\right\|^3}\right\| < \frac{N-1}{\gamma^3}\|z_i - z_c m\| \tag{89}$$

and hence

$$\dot{V}_i < -k_f\|\dot{v}_i\|^2 \tag{90}$$

which proves the given theorem.

V. Implementation of X-Swarm Theory

Two conditions to create a X-Swarm is to find two positive variables which satisfy the following inequalities

$$\begin{aligned} &\succ \quad d_{ij} \geq \gamma + \delta \\ &\succ \quad \left\|\frac{z_i - z_{cm}^i}{z_i - z_{cm}}\right\| < \left(1 + \frac{\delta}{\gamma}\right)^3 \end{aligned}$$

It is possible to check the X-swarm conditions for a swarm with the $min(d_{ij})$ and $max \left\| \frac{z_i - z^i_{cm}}{z_i - z_{cm}} \right\|$. Lets define the variable of $X_{sw_{min}}$ and $X_{sw_{max}}$ as following:

$$X_{sw_{max}} := max \left\| \frac{z_i - z^i_{cm}}{z_i - z_{cm}} \right\| \quad X_{sw_{min}} := min(d_{ij}) \tag{91}$$

To satisfy the X-swarm inequalities, it is possible to write

$$X_{sw_{min}} = \gamma + \delta \quad X_{sw_{max}} = (1 + \frac{\delta}{\gamma})^3 \tag{92}$$

by using the above equations it is possible to calculate the variables of γ and δ can be calculated with,

$$\gamma = \frac{X_{sw_{min}}}{\sqrt[3]{X_{sw_{max}}}} \tag{93}$$

and

$$\delta = X_{sw_{min}} - \frac{X_{sw_{min}}}{\sqrt[3]{X_{sw_{max}}}} \tag{94}$$

if the calculated γ and δ are both positive, than the X-swarm conditions are satisfied. Otherwise, the main approach is to increase the variable gain of k_m which determines the magnitude of the repulsive forces between agents to increase the minimum distance between the agents in the swarm, $X_{sw_{min}} := min(d_{ij})$

On the other hand, the theorem related with the convergence of member agents of an X-Swarm to the z_cm is satisfied under the condition of $\frac{k_a}{k_m} > \frac{N-1}{\gamma^3 l(C)}$. Another approach is to increase the k_a gain to make the related inequality be satisfied to force the agents which are outside of the desired shape to the center mass of the formation geometry.

IV. Shape Partitioning Methods

Shape partitioning methods have basically reduced down the formation control problem into two subproblems. The first part of the solution is to partition the desired formation shape into potential goal states according to the agent types to cover the desired formation shape homogenously. There are two different solutions to the shape partitioning problem are presented in this thesis work, bubble packing method and randomized fractals method. The second part of the solution is the decision process of the agents to select these goal states continuously to minimize the energy consumption of the swarm. During this decision process, the cost of reaching different goal states will be the main criteria for each agent. It is obvious that each agent will try to choose a goal state with minimum cost according to its position and the orientation in the environment. But the cases in which two or more agents are willing to reach the same goal point must be handled to optimize the overall utility of the swarm.

Shape partitioning methods have an approach to direct the agents to the goal states in which they have assigned instantly to minimize the energy consumption, but it is important to keep the agents together in the swarm while moving on a trajectory towards the formation shape, to reduce the Lost agent events described in Section-xx. For this purpose, attractive forces which directs the agents to the center of the formation shape under X-swarm consideration is added to the agents' control signal as an additive term.

I. Determining the Potential Goal States

I.1 Bubble Packing Method

Bubble packing method is widely used in mesh generation problems. It basically depends on covering a curve, surface or a volume with a proper number of bubbles by packing them tightly which mimic a Voronoi diagram, from which a set of well-shaped Delaunay triangles and tetrahedra can be created by connecting the centers of the bubbles[27]. The algorithm places the bubbles with their initial conditions in the surface and apply them interbubble forces which imitates the Van der Waals forces between the moleculer bonds to distrubute the bubbles homogenously. Here, the main idea is to generate a mesh for a surface with identical bubbles to mimic a regular Voronoi diagram with the vertices represented by the centers of these bubbles. On the other hand, adaptive population control methods are developed to increase the number of bubbles to fill the gaps in the shape and to remove the excess bubbles which are overlapping with each other and the shape boundaries. Since the numbers and the the radius of coverage circles which is defined at Section-xx for the agents are predetermined in our formation control problem, the general bubble meshing algorithm have to be adopted to meet the requirements in shape partitioning in formation control. The basic approach is to represent the agents in the swarm as bubbles with the radius of their coverage circles and create a mesh by using these bubbles.

I - Initial Placements of the Bubbles The initial bubble placements are important because it will greatly reduce the convergence time of the partitioning process. In this work, the bubbles are initiated close to the center of the formation shape randomly. The algorithm for initial bubble placements is provided as follows:

Data: Set of Bubbles, Desired Formation Shape
Result: Initial Placements of the Bubbles
Initialize free configuration space C_{free} as the desired formation shape
for <*Bubbles of i*> **do**
    *Calculate the free configuration space C_{free} for bubble i;
    *Put the Bubble i to the closest point to formation center z_c in the free configuration space i ;
    *Add the agent i into the desired formation shape as an obstacle ;
end

Algorithm 1: INITIALIZE_BUBBLE_POSITIOINS

The term free configuration space C_{free} will be analyzed in detail in Section -xx. An execution of the Algorithm-1 is illustrated in Figure -xx below.

Giris formation shape ve bubble lar, cikista merkezde toplanmis bubblelar olan bir sekil koyalim

II- Bubble Meshing Process

The bubbles are distributed homogenously with this process under two kinds of forces, interbubble forces and shape repulsive forces. The interbubble forces are proximity-based forces so that a system of bubbles is in equilibrium when bubbles are distributed over the whole formation shape. The implemented force equation is given

$$f_i(l) = \begin{cases} al^3 + bl^2 + cl + d & \text{when } 0 \leq \mathrm{l} \leq l_0 \\ 0 & \mathrm{l} > l_0 \end{cases} \tag{95}$$

where l is the distance between the centers of the related bubbles. Makaleden interbubble force seklini alalim ya da biz cizelim

The shape repulsive forces have the same characteristics with the repulsive artificial forces

discussed in Section -xx. The representation of shape repulsive forces for the desired formation shape C with the points of $z_k = (x_k, y_k)$ the complex plane::

$$f_r(X_i) := \oint_C \left[\frac{\alpha X_i - z}{\|\alpha X_i - z\|^3} \right] \|dz\| \tag{96}$$

where k_r is the variable gain for the repulsive forces. The representation of the repulsive forces on agent i on $z_i = (x_i, y_i)$ with the points of $z_k = (x_k, y_k)$ of formation shape in the complex plane:

The bubbles are distributed homogenously under the influence of these two forces when they get an equilibrium state in which the total net forces acting on individual bubbles reaches zero. The final equilibrium states of the bubbles determines the potential goal states of the agents in the swarm to cover the formation shape.

Buraya initial pozisyonlardan nihai pozisyona gelinmis goal state seklini gosteren bir resim koyalim. (Iteration lar ile)

I.2 Randomized Fractals Method

Randomized fractals method are used to cover surfaces or volumes randomly with fractals. The main idea is to fill the fractals with the areas determined by the rule of [26] :

$$A_i = \frac{A}{\zeta(c, N)(i + N)^c} \tag{97}$$

where A is the total are to cover and A_i is the area of the i^th fractal. The parameters of c and N can be chosen to implement different changes on the fractals areas with the increasing number of iterations with $c > 1$ and $N > 0$. Here ζ is the Hurwitz function defiend by

$$\zeta(c, N) = \sum_{i=0}^{\infty} \frac{1}{(i + N)^c} \tag{98}$$

It is possble to write,

$$\sum_{i=0}^{\infty} A_i = \sum_{i=0}^{\infty} \left(\frac{A}{\zeta(c, N)(i + N)^c} \right) \tag{99}$$

which tells us the sum of the all areas A_i is the total area of A and the algorithm is space filling. This approach implements the fractals infinitely by reducing the areas in accordance with the Equation -xx to the desired formation shape randomly. Paperdan koymadigimiz bir sekil koyalim

Since the areas and the number of the agents, which will be represented with their coverage

circles, are predetermined in our work, it is possible to adopt this algorithm as follows:

Data: Set of Coverage Circles, Desired Formation Shape
Result: Potential Goal States
Initialize free configuration space C_{free} as the desired formation shape
for <*Coverage Circles of i*> **do**
  *Calculate the free configuration space C_{free} for circle i;
  if $C_{free} \neq Null$ **then**
    *Put the circle i randomly in C_{free} ;
    *Add the agent i into the desired formation shape as an obstacle ;
  else
    *Break and warn the operator to increase the size of formation shape
  end
end

Algorithm 2: RANDOMIZED_FRACTALS_ALGORITHMS

II. Mesh Quality Measurement

Since we have two different solutions to the shape partitioning problem, it is useful to define a method to compare the performances of these algorithms. One of the criterion of mesh diagrams is topological mesh irregularity [27-29]. This parametere is defined by :

$$\epsilon_t = \frac{1}{n}\sum_{i=0}^{n} |\gamma_i - D| \tag{100}$$

where

$$D = \begin{cases} 6 & \text{for triangles in Voronoi Diagram} \\ 12 & \text{for tetrahedras in Voronoi Diagram} \end{cases} \tag{101}$$

and γ_i represents the degree, or the number of neighboring nodes connected to the i^th interior node, and n represents the total number of interior nodes, i.e. the number of bubbles or fractals. It is obvious that the topological mesh irregularity vanishes when all nodes have D neighbors, but it is almost not possible in practical application.

Burada 3=5 komusus olan bir node u ucgen ya da tetra voronoi diagrami ile verelim

Another metric to evaluate the quality or the performance of the mesh is the geometric irregularity defined as[27]:

$$\epsilon_g = \frac{1}{m}\sum_{i=0}^{m}(A - \frac{r_i}{R_i}) \tag{102}$$

where

$$A = \begin{cases} 0.5 & \text{for triangles in Voronoi Diagram} \\ \sqrt{2/11} & \text{for tetrahedras in Voronoi Diagram} \end{cases} \tag{103}$$

and m represents the number of nodes, r_i is the radii of inscribed circle of Voronoi cell belonging to node i and R_i is the radii of the circumscribing circle of Voronoi cell belonging to node i

Burada bir ucgen bir tetraheadral icindeki disindaki daIRELERLE voronoi verelim

III. Decision Process on Goal States

In the first part of the work,Section-xx Shape Partitioning, the formation control problem is reduced down to a problem in which every agent is expected to decide individually where to

position in a given set of possible goal states $g_i \in G$. During this decision process, the cost of reaching different goal states will be the main criteria for each agent. Given goal states and cost values to these goal states, each agent must individually decide where to position in the formation. This process must be held to optimize the utility of every agent with a collaboration. It is obvious that some of the agents may want to choose the same goal point to reach, so the swarm must reach a global consensus on target points and conflict cases must be handled. The main approach to provide a solution for this problem is to make each agent to calculate the costs of its own to the goal states and then create a global consensus with the other agents to minimize the overall energy consumption of the swarm while achieving a formation shape. In this work, the cost of reaching a goal state is defined with the minimum shortest path in the environment. A visibility graph based approach is used to calculate the shortest possible path to a goal states by taking into account the obstacles in the environment. The assignment process of the agents to the goal states which will be minimize the overall cost, is handled with the help of Munkres (Hungarian) algorithm.

III.1 Free Configuration Space

Since the shortest path is defined in the free configuration domain, each agent must calculate its free space by taking the obstacles in the environment into the account.

Assume an environment with set of obstacles $S = \{P_1, P_2, ..P_t\}$. Configuration for agent i can be described with the position of the center of its coverage circle with $R = \{x_i, y_i\}$. Configuration space of i th agent is the environment itself and represented by $C(R_i)$. This configuration space is composed of two subspaces; free configuration space and forbidden configuration space of agent i.

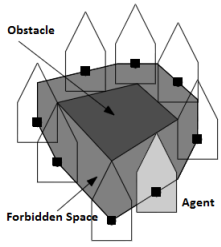
$$C(R_i) = C_{free}(R_i, S) + C_{forb}(R_i, S) \tag{104}$$

The configuration space $C(R_i)$ of agent i is the environment itself. If the forbidden space is calculated with Minkowski Sum method, free configuration space can be derived simply by extracting the forbidden space from the environment. Let a single obstacle is described with a point set of S_1 and the agent is described with a point set of S_2. The Minkowski sum of these two sets $S_1 \subset R^2$ and $S_2 \subset R^2$ can be calculated with the following,

$$S_1 \oplus S_2 := \{p + q : p \subset S_1, q \subset S_2\} \tag{105}$$

where $p + q$ denotes the vector sum of the vectors p and q.

Figure 1: *Forbidden Space Caused by an Obstacle*



Forbidden space for agent i, $C_{forb}(R_i, S)$ is the sum of the forbidden spaces calculated for each obstacle in the environment.

III.2 Visibility Graphs and Dijkstra's Algorihtm

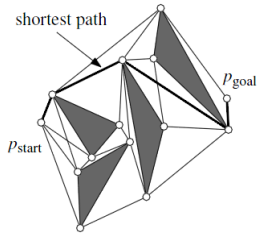
It is obvious that the shortest path between two points in the environment must lie on the free configuration space, C_{free} for each agent. A constraint for the shortest path is given as follows[30] :

> The shortest path between p_{start} and p_{goal} among a set S of augmented polygonal obstacles consists of the arcs of the visibility graph $\gamma_{vis}(S^*)$ where $S^* := S \cup \{p_{start}, p_{goal}\}$

A visibility graph,$\gamma_{vis}(S^*)$, is a graph which is set of interior nodes representing the vertices of the set of obstacles, S, in the environment and edges which represents visible (which are not crossing and interior region of an obstacel) connections between these nodes.

Consider set of obstacles in the environment is augmented with the Minkowski Sums described in the Section-xx . Let these set of augmented polygonal obstacles represented with $S_i \subset S$.

Figure 2: *Shorthest Path from an initial state to a goal state*



Algorithm to calculate the visibility graph of S^*,

Data: Set of Vertices Included in S^*
Result: Visibility Graph of S^*
Initialize a graph $\gamma = (V, E)$ where V is the set of al vertices of the polygons in S and $E = \oslash$
for <*all vertices* $v \subset V$> **do**
  W = VISIBLEVERTICES(v,S);
end

Algorithm 3: VISIBILITYGRAPH(S^*)

where $VISIBLEVERTICES(v, S)$ algorithm checks whether line segments drawn from v to all vertices in S is intersecting an interior area of any obstacle in the environment. With the help of this $VISIBILITYGRAPH$ algorithm $SHORTESTPATH$ algorithm can be defined as follows.

Data: A set S of disjoint polygonal obstacles, and two points p_{start} and p_{goal} in the free space.
Result: The shortest collision-free path connecting p_{start} and p_{goal}
1) Assign γ = VISIBILITYGRAPH(S^*)
2) Assign each arc (v, w) in γ_{vis} a weight, which is the Euclidian length of segment vw
3) Use Dijkstra's algorithm to compute a shortest path between p_{start} and p_{goal} in γ_{vis}

Algorithm 4: SHORTESTPATH

Dijkstra's algorithm computes the shortest path between two nodes in graph with k arcs, each having a non-negative weight. It is a tree search algorithm and used to calculate the shortest paths between nodes in a graphs, with the help of weighted edges between nodes. The time complexity of the original algorithm is $O(n^2)$ where n is the number of the nodes in the graph. With the usage self balancing binary search tree, the algorithm requires $O(k + nlogn)$ time in the worst

case. The algorithm for the Dijkstra's is given as follows:

Data: γ_{vis} , *source_node*
Result: Shortest Distance to Any Node from *source_node* in γ_{vis}
for <*each vertex* $v \subset \gamma_{vis}$> **do**
  Distance[v] := ∞ ;
  Previous[v] := $undefined$;
end
Distance[*source_node*] :=0 ;
Q:= The set of all vertices $v \subset \gamma_{vis}$;
while <$Q \neq null$> **do**
  u:= Node in Q with smallest distance to *source_node*;
  remove u from Q;
  for <*each neighbor v of u*> **do**
    alt:= Distance[u] + Cost Between u and v nodes;
    if *alt<Distance[v]* **then**
      Distance[v] :=alt;
      Previous[v] :=u;
    end
  end
end
return Previous[];

Algorithm 5: DIJKSTRA'S ALGORITHM

In algorithm above $Distance[x]$ function call, calculates the total cost from the *source_node* to the x vertex, and $Previous[x]$ function call, returns the previous node in optimal path from *source_node* . In our work, the weights of the edges in γ_{vis} , are calculated with the Euler distance between nodes in the environment.

III.3 Collaborative Decision Process of Final Goal States

In Section 2 and 3 possible routes and costs to the goal states are calculated with the help of Visibility Graphs and Dijkstra's algorithm for each agent. It is obvious that each agent will try to choose a goal state with minimum cost according to its position and the orientation in the environment. But the cases in which two or more agents are willing to reach the same goal point must be handled by optimizing the overall utility of the swarm. To minimize to overall cost of whole swarm while achieving a formation shape, Hungarian algorithm which is a combinational optimization algorithm that solves this assignment problem is used. To implement this algorithm, a complete bipartite graph $G = (S, T, E)$ with $n \in S$ agents and $t \in T$ goal points is constructed.In this graph, each agent have a cost which is defined by the shortest path to the destination in the environment for different goal points.

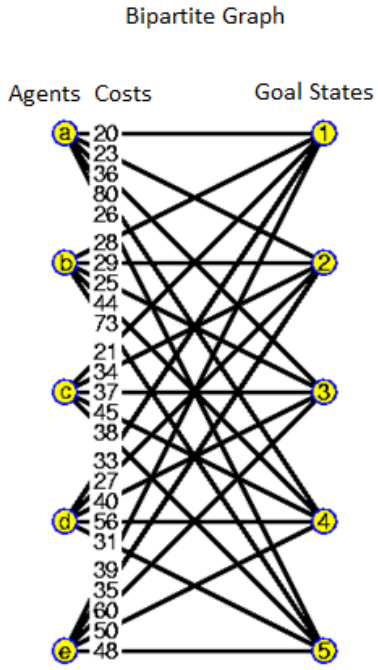


Figure 3: *Sample Bipartite Graph Used in Assignment Problem*

A cost matrix *C* is defined to implement the Hungarian algorithm . The dimensions of the cost matrix is *nxm* in which each element represents the cost of assigning the goal state *m* to the agent *n*. Since we have equal number of agents and goal states, the cost matrix will be a square matrix

with *nxn* or *mxm*. The algorithm for the assignment process as follows:

Data: Cost Matrix , *C*
Result: Assignment Array of Agents to Goal States
Label1 ;
for *Each Row, R, in C* **do**
| Find the smallest element and subtract it from every element
end
Label 2 ;
if *A Column,K, contains more than one zero* **then**
| Repeat Label1 for each column, *K*
end
Label 3 ;
Select element in columns for which a distinct minimum weight has been determined and add to solution
Label 4 ;
If it is not possible to reach the full solution, flag rows without solutions. Flag all columns in flagged rows that contain a zero. Flag all rows with a previously determined solution in previously flagged columns.
Label 5 ;
From elements remaining in flagged rows and unflagged rows, determine the element which has smallest value and assign this value to γ. Subtract γ from every unflagged element and add γ to every element that has been flagged twice.
Label6 ;
Goto Label3 until full solution has been achieved.

Algorithm 6: HUNGARIAN ALGORITHM

IV. Control System Design for Individual Agents

Shape partitioning methods will provide the goal states for a desired formation shape and agents will be assigned to those goal states to minimize the overall energy consumption of the swarm. A control system which will guide the agents to reach these goal states must be designed for each agent individually. Since the environment is dynamically changing with lots of mobile agents, it is very probable to have different assignments to these goal states at each iteration of formation control. On the other hand, the collision with the obstacles and the other agents at the environment must be prevented due the requirements presented in Section-xx. The control system must react to these dynamic requests. A velocity controller with a large bandwith is designed at the inner loop of the control system and the outer loop is composed with a velocity setpoint generator. The block diagram for the proposed controller structure is presented in Figure -xx.

Genel blok diyagram buraya eklenecek

Velocity setpoint generator provides instant setpoints for the inner loop based on the current position of the agent and the desired goal state position. This loop calculates the amplitude of the velocity setpoint proportinal with the euclidian distance of agent to the desired goal state. The direction of this velocity setpoint vector has a bearing angle of the line segment drawn from the agent to the goal state. This generator saturates the amplitude of the setpoint vector with 0.5 [m/sec] to prevent the agents' travelling in the environment so fast.

Amplitude ve bearing sekli buraya konacak

The inner loop tis designed to track the velocity setpoint provided by the generator and it has a state feedback structure. The gains for the feedback states are calculated with an LQR controller. A simple mass, damper type second order linear system is used to provide a model for the controller.

The translational friction force is assumed to be linear with the velocity of each agent. Different mass and friction coefficients for heterogenous mobile robots are used in control system design. To track the desired velocity setpoint, the model of the system is augmented with an artificial error state,

$$\begin{bmatrix} \dot{v} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} -b/m & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ e \end{bmatrix} + \begin{bmatrix} 1/m \\ 0 \end{bmatrix} F_{net} \quad and \quad y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ e \end{bmatrix} \tag{106}$$

where b is the linear friction force coefficient and m is the mass, v is the linear velocity of the agent and e is the augmented error state which is the integral of the velocity state. Here the v state is the stabilizing part of the controller and the e error state is the tracker part of the controller. The state feedback gain, K, which minimizes the quadratic cost function of

$$J = \int_{t_o}^{t_1} (x^T Q x + u^T R u) dt \tag{107}$$

is calculated with help of $'lqr'$ function of MATLAB for the given system in $106 - degisebilir$ with the gain matrices of

$$Q = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix} \quad and \quad R = r_1 \tag{108}$$

the parameters for the controller design are determined with the approach presented below,

$$q_1 = \frac{1}{t_1 (x_{1max})^2};\ q_2 = \frac{1}{t_2 (x_{2max})^2};\ and\ r_1 = \frac{1}{(u_{1max})^2}; \tag{109}$$

Here t_i is the desired settling time for x_i which is determined as 1.5 seconds for the velocity and 0.001 seconds for the integral state. The statement of x_{imax} represents the expected maximum value of the state i , which is defined 1 [m/sec] for the velocity state and 4 [m] for the error state. $u1max$ represent the maximum allowable input signal which is defined 3 [N]. The structure of the inner loop is illustrated in Figure-xx

Inner loop sekli gelecek

The error between the reference and the velocity feedback is integrated and multiplied with the error gain, and the velocity state is multiplied with the velocity gain. These two components generate the total control input of the system.

Step response buraya gelecek

According to the Figure -xx, the settling time for the inner loop is around 0.5 seconds for a step input. This response characteristics is important since the system cannot react to the setpoints changing faster than 0.5 seconds, so the formation control loops in which the new goal state assignments are handled must be executed with 2Hz frequency.

V. RESULTS AND DISCUSSION

In this section the results of the simulation are analyzed and discussed in detail for the LPS system design and Formation Control system design. Three different methods of Formation control system which are presented in Section-xx are evaluated and compared with each other. Simulations are handled in the environment of Gazebo simulator with a swarm of 50 agents which includes three different types of robots. The main system frequncy is determined as 2Hz due to the bandwith of the controllers which is discussed in Section-xx.

I. Simulation Environment

Block diagram of the simulation environment is illustrated in Figure -xx.

Environment sekli buraya konacak.

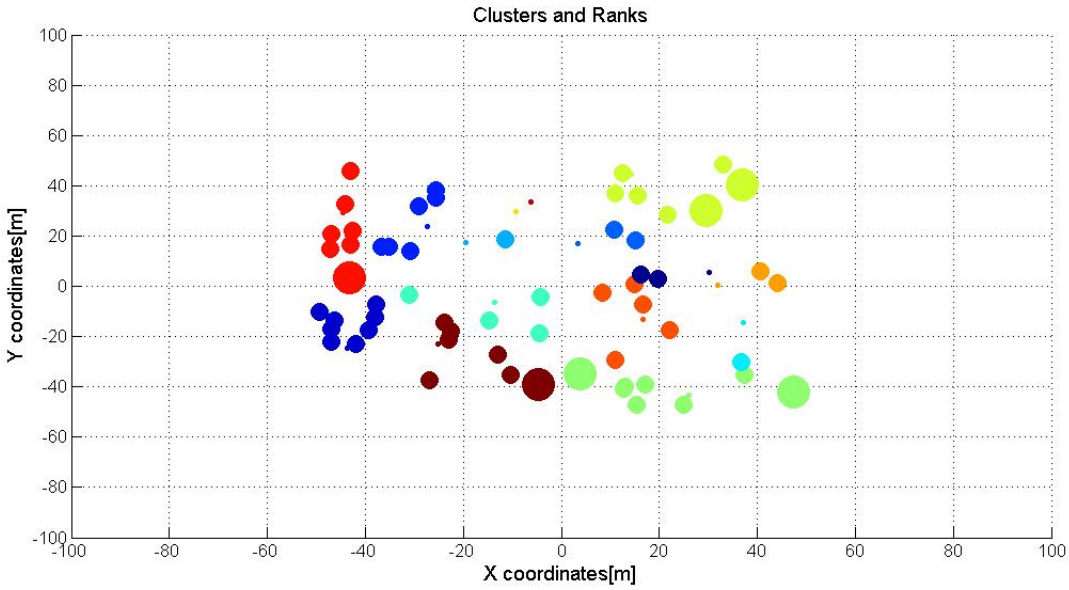
Gazebo is an open source multi robot simulator developed by OSRF(Open Source Robotics Foundation). It has multiple physics engine ODE (Open Dynamics Engine), Bullet, DART(Dynamic Animation and Robotics Toolkit) and Simbody. It can execute simulations with multiple agents in an environment that is fully determined by the user. The dynamics and the physical properties of the robots are determined by the user. It supports to integrate with ROS (Robot Operating System) Shape partitioning algorithms are executed in Matlab environment and potential goal states are published over a network socket. A plugin working in Gazebo environment listens the packets sent by Matlab and parse the data for each agent. Agents have their own plugins that listens goal states and execute their decision process algorithms. These plugins also have controller algorithms to make the agents reached to the goal states they have assigned. Dynamics of the agents are handled in the physics engines of the Gazebo simulator. The state vectors of the agents are published with a plugin from the Gazebo simulator and these state datas are used in the shape partitioning algorithms together with the formation shapes demanded by the user. The environment is chosen as a rough 3D territory and each agent have different interactions with the environment with different mass and friction characteristics.

Uc farkli agent seklini ve environment i gosteren Gazebo sekli

II. LPS Design

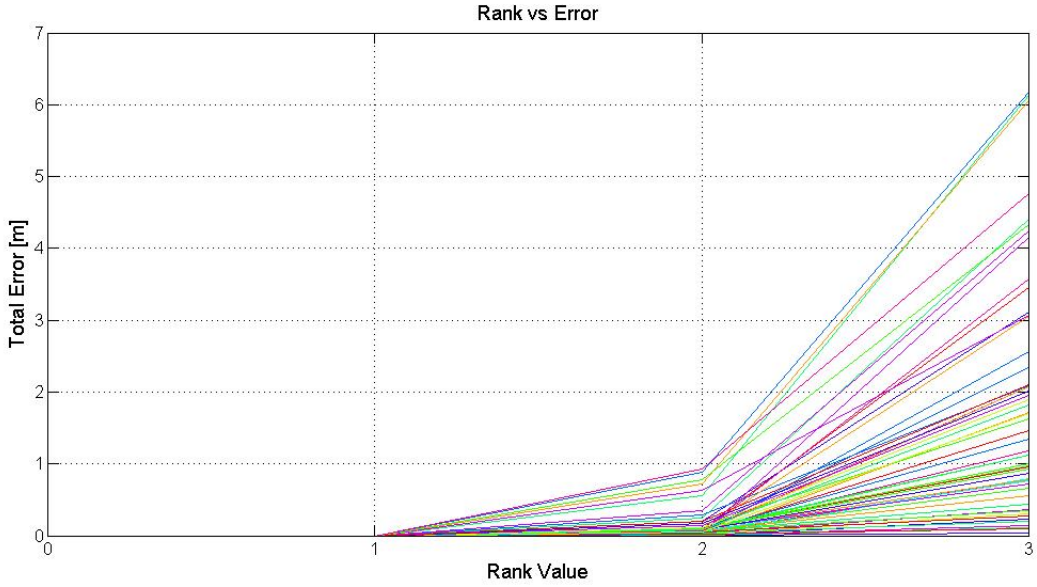
In the LPS design, the main aim is to design an architecture in which every agent can localize itself with the help of the position agents which have positioning sensors on their boards. With the help of DSDV algorithms, each individual agent assign itself to the cluster of position agent with minimum number of hops in the network which is discussed in Section-xx. A sample output of this algorithm at an instant time during simulations is illustrated in Figure-xx.

Figure 4: *Cluster Assignments of the Agents*



Agents with the same colors are assigned to same clusters at this instant time and their size represents the rank number of each agent at that cluster which is determined with the number of hops. It is important to notice that agents are assigned to the clusters of the position agents with minimum number of hops rather than the minimum distance since it is expected to have greater errors on position data with the increasing number of ranks because of the cumulative error effect. This assumption is supported with the simulation experiments and results are presented in Figure-xx. The agents which are not directly get into trilateration process with the position agents (i.e. agents with rank greater than 2) have to localize themselves with the help of their neighbor with lower ranks,thus their position data will include the errors of the localization process of their neighbors. This situation is illustrated in Figure-x, total error is increasing cumulatively with the number of rank.

Figure 5: *Total Error on Position Data with Respect to Ranks*



The state propogation period which is discussed in Section-xx is determined as 2Hz as equal to the main system frequency since the new position datas for the formation control system are required to be determined with this rate. As discussed in Section-xx, a localization timer is proposed to handle the DSDV algorithm and trilateration process which will provide a greater execution period than the propogation process of the states with the help of inertial measurements. It is obvious that the state variables will be drifting with the increasing time because of the measurement and process noise in propogation process unless they are not corrected with the external measurements, so it is possible to determine a minimum execution period of this trilateration process to satisfy an error band in the state variables.

Figure 6: *Total Error with Localization Timer Period of 3 Seconds*

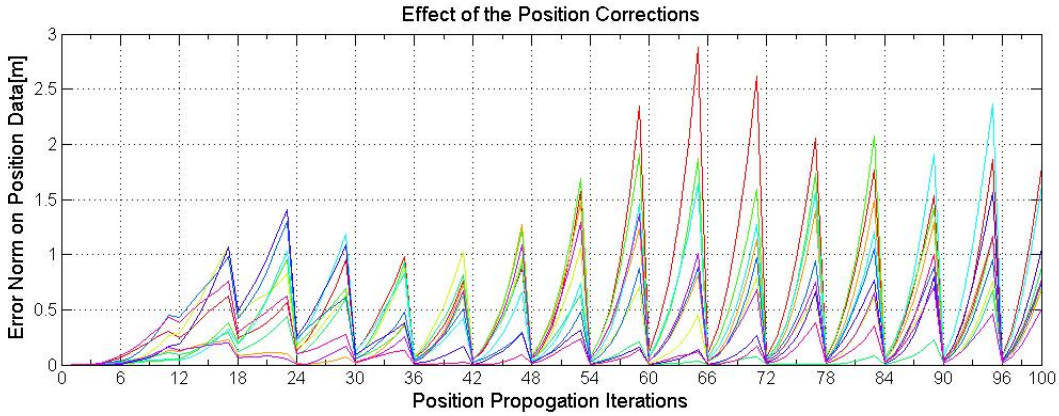


Figure 7: *Total Error with Localization Timer Period of 5 Seconds*

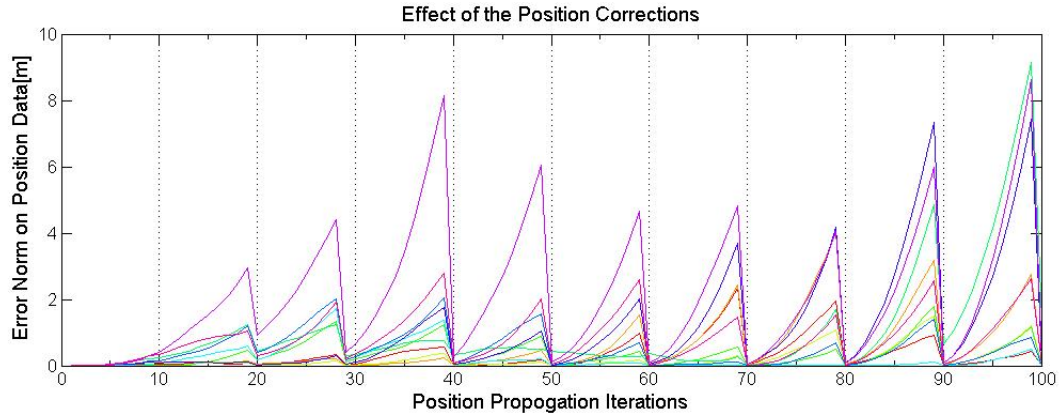
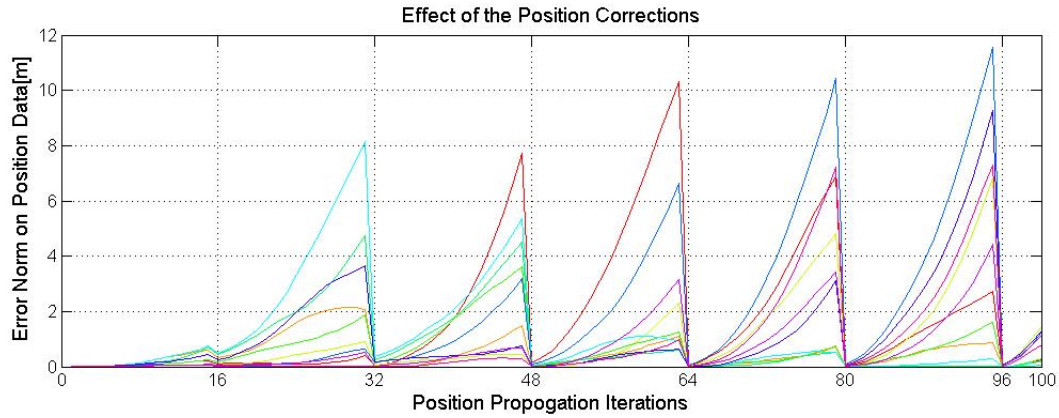


Figure 8: *Total Error with Localization Timer Period of 8 Seconds*



It is clear that the total error norm on position datas are decreased dramatically with the localization period which equals to 6 iterations for 3 seconds period, 10 iterations for 5 seconds

and 16 iterations for 8 seconds period. The peak values on the error norms are always observed at the iterations just before the localization process. It can be concluded that the peak values on the error norms are greatly related with the localization period and they have greater values with the increasing number of this period. The maximum error norm is below 3 meters with localization period of 3 seconds, below 10 meters with localization period of 5 seconds and below 12 meters with localization period of 8 seconds. It is decided that the error norm of 3 meters is the maximum tolerable value for the formation control system and the localization period is determined as 3 seconds.

The performance of the LPS design is tested in simulations with different conditions in which swarm is propogating to a desired formation shape or agents are keeping their positions in a given formation shapes. In both cases, it is observed that the position datas of the agents are drifted with an increasing error between two sequential localization process. A sample case for this situation is illustrated in Figure -xx

Figure 9: *Positions of the Agents Before Localization Process*
(Red Circles: Estimated Positions ; Blue Circles: Real Positions)

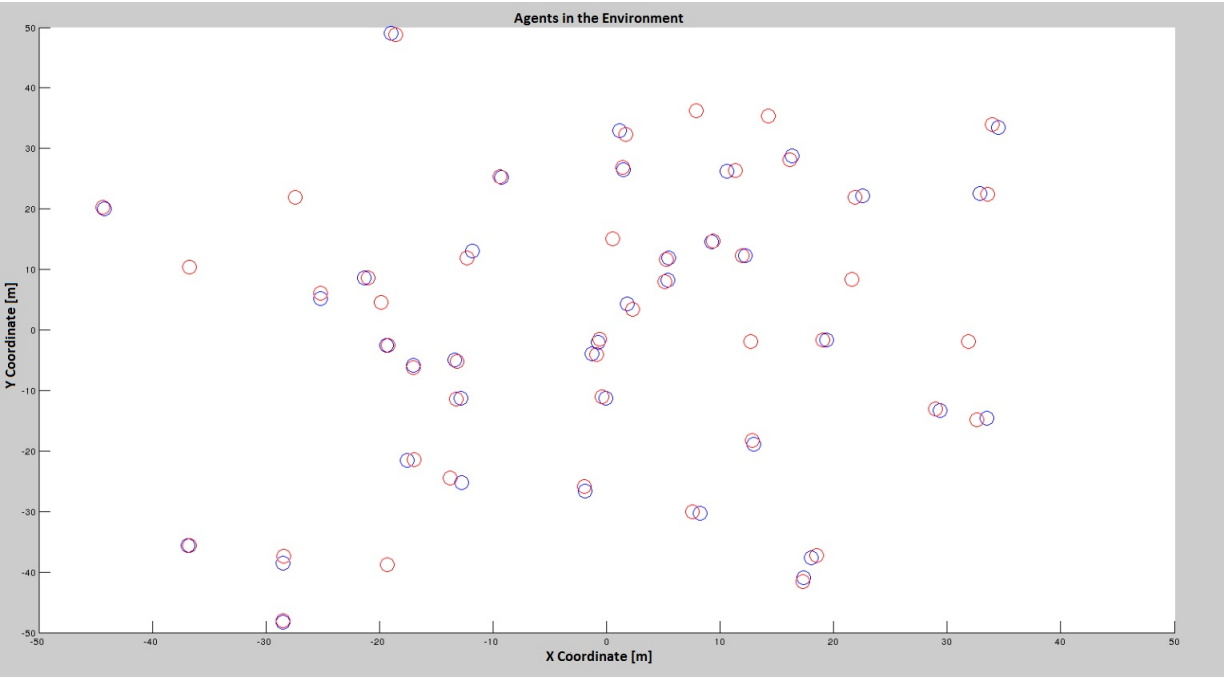
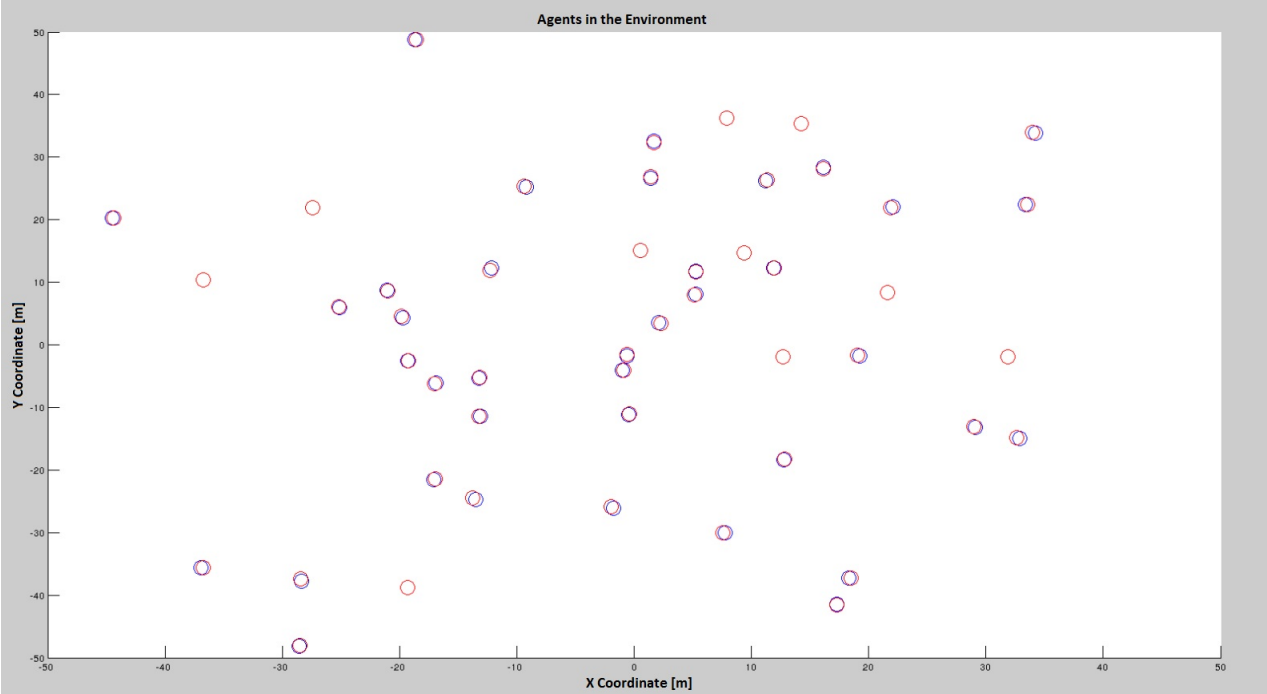


Figure 10: *Positions of the Agents After Localization Process*
(Red Circles: Estimated Positions ; Blue Circles: Real Positions)



Handling procedures of the lost agents are presented in Section-xx. Agents which do not have three neighbors around itselves will get into the 'Lost' mode, and if they miss three localization process they will get into 'Come Home' mode. When an agent have 'Come Home' mode, it will try to reach the center of formation to increase the possibility to meet some neighbors to localize itself as discussed in Section-xx. A simulation result in which two agents cannot have three neighbors around themselves and get into 'Lost' mode and 'Come Home' modes is presented in Figure -xx and Figure-xx respectively.

Figure 11: *Agents will be in 'Lost' Mode When They Do Not Have 3 Neighbors*

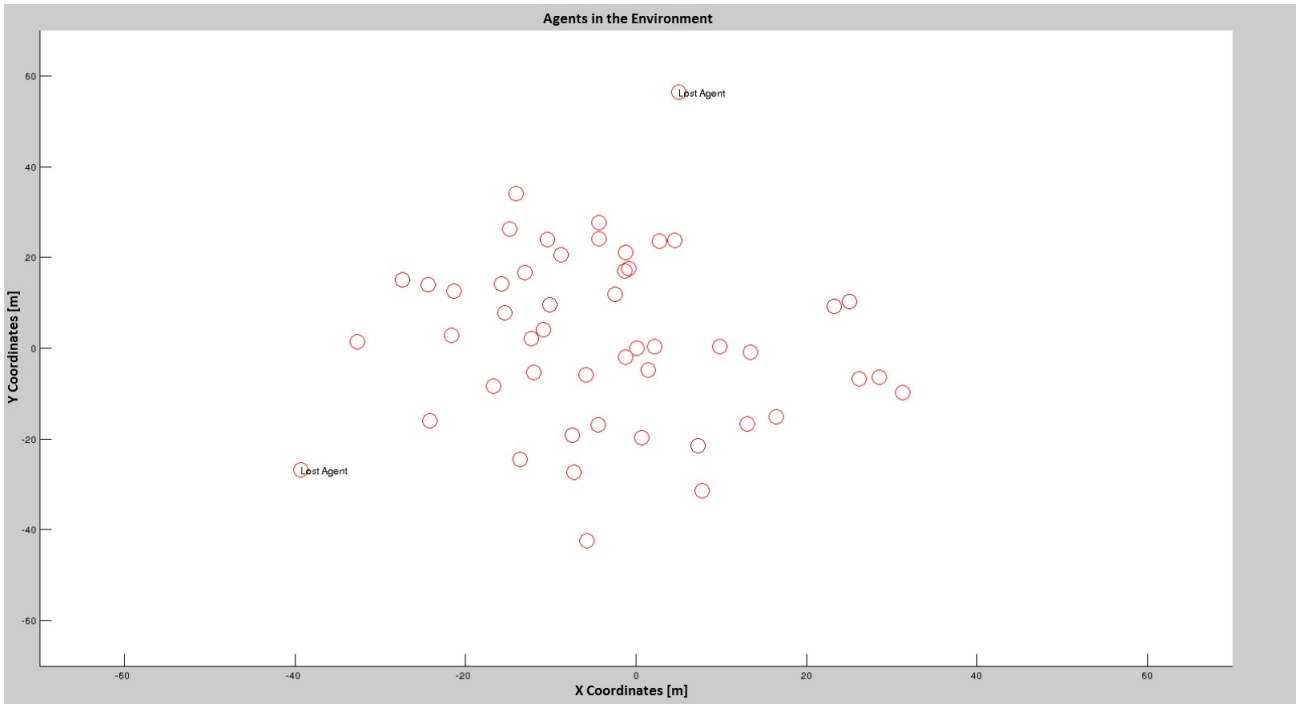
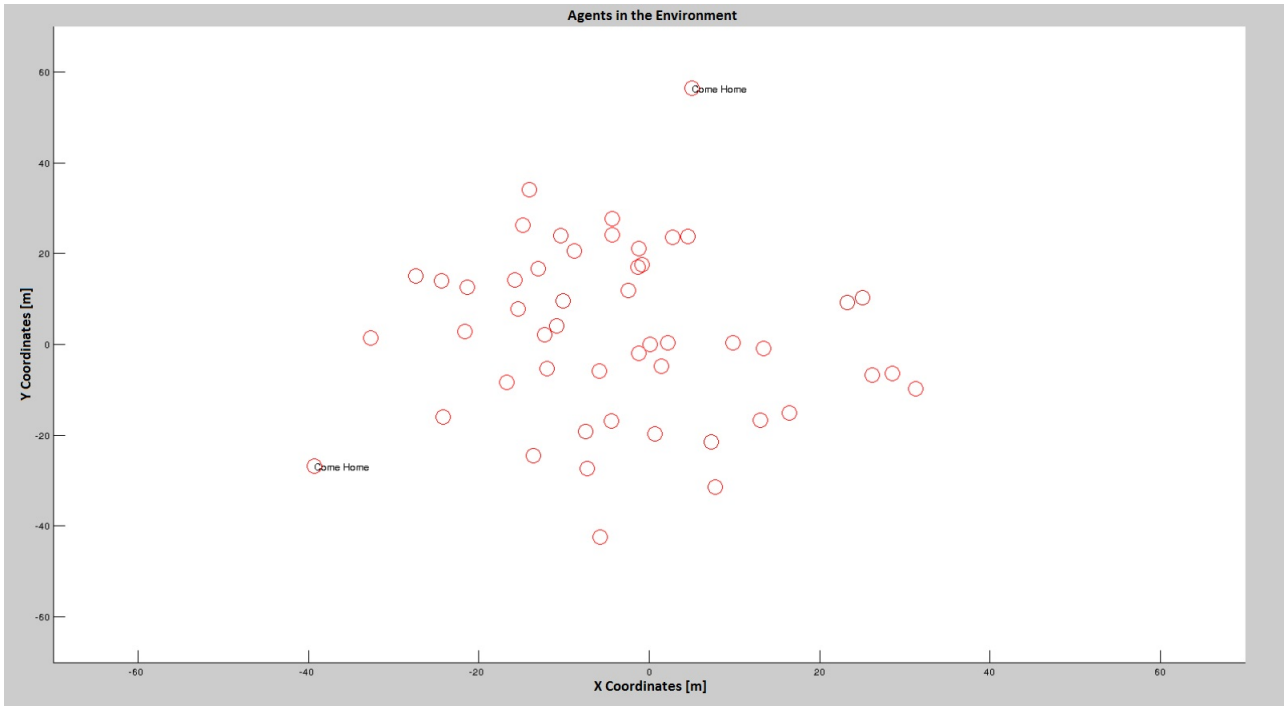


Figure 12: *Agents will be in 'Come Home' Mode When They Miss Localization Process for 3 Times*



III. FORMATION CONTROL SYSTEM

Three dÄśfferent approaches of which the details are presented is Section-xx are evaluated according to their settling times, mesh qualities and energy consumptions.

III.1 Mesh Qualities

Mesh qualities is a measure of how the agents homogenously distributed while covering the desired formation shape. Basically, two different types of quality measurents defined in Section-xx are used to evaluate the performance of the formation control methods, topological mesh irregularity ϵ_t and geometrical mesh irregularity ϵ_g . Monte Carlo simulations with 1000 iterations are handled for the same formations shapes with different initial conditions of the agents in the environment. Sample outputs for three different types of formation control algorithms are illustrated in the following Figures

Figure 13: *Formation Shape 1 with Artificial Forces Methods:*$\epsilon_t = 2.1$ *and* $\epsilon_g = 0.32$

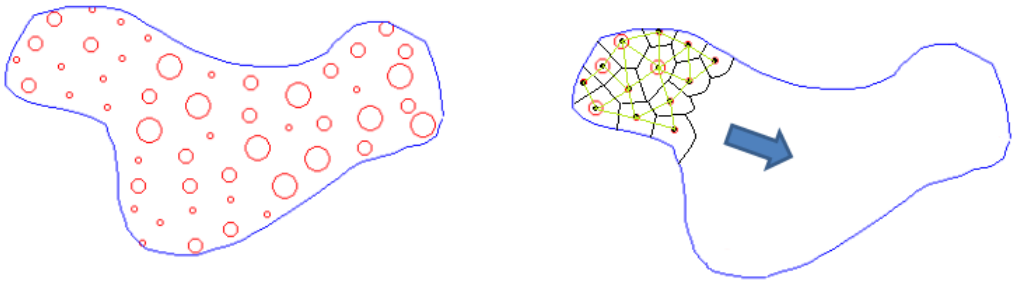


Figure 14: *Formation Shape 2 with Artificial Forces Methods:*$\epsilon_t = 2.6$ *and* $\epsilon_g = 0.4$

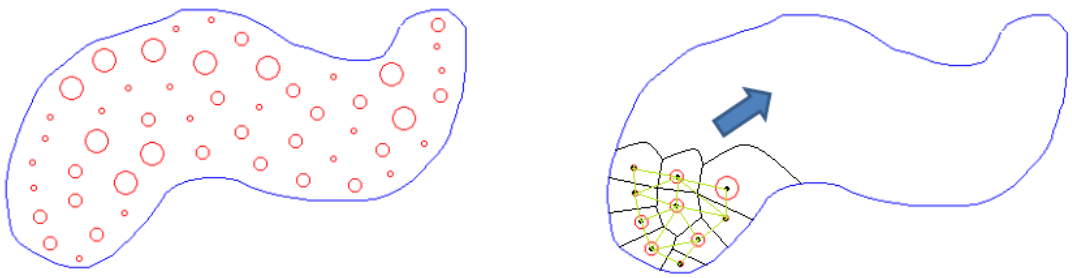


Figure 15: *Formation Shape 1 with Bubble Packing Method:*$\epsilon_t = 2.1$ *and* $\epsilon_g = 0.24$

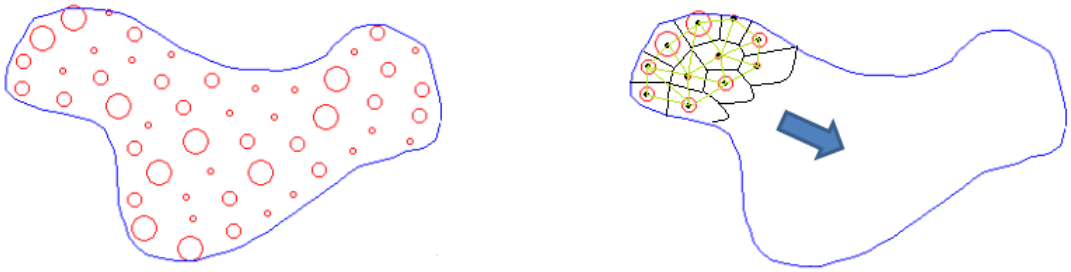


Figure 16: *Formation Shape 2 with Bubble Packing Method:*$\epsilon_t = 2.3$ *and* $\epsilon_g = 0.28$

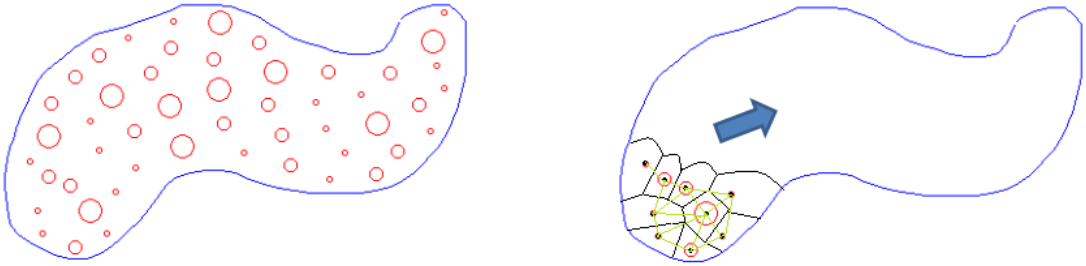


Figure 17: *Formation Shape 1 with Randomized Fractals Method:*$\epsilon_t = 2.7$ *and* $\epsilon_g = 0.65$

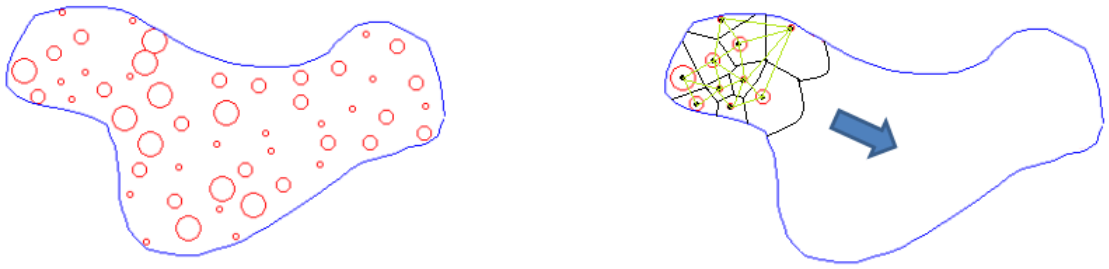
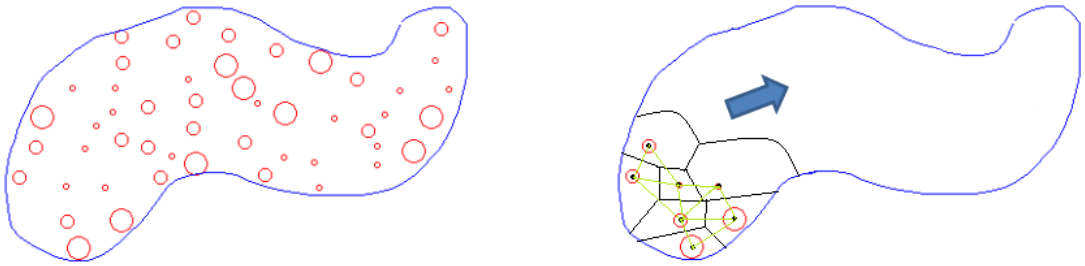


Figure 18: *Formation Shape 2 with Randomized Fractals Method:*$\epsilon_t = 3.1$ *and* $\epsilon_g = 0.62$



It is obvious that Randomized Fractals method have the worst mesh performance due the its randomized assignment of goal states into the formation shape. Bubble Packing and Artificial Forces methods give similar results since they have an analogy in their approaches of implementing inter member/bubble forces which makes the agents distributed in the formation shape more homogenously. Artificial force method applies this intermember forces directly to the agents in real time while Bubble Packing method use this kind of force on artificial bubbles to partition the shape into goal states. Monte Carlo simulations with 1000 iterations for both formation shapes are handled and the results are illustrated in Figure -x1 -x2 . Randomized Fractals method have greater mean values for both topological and geometrical mesh irregularities as expected. Bubble Packing and Artificial Forces methods have similar performances on Mesh quality.

Figure 19: *Formation Shape 1 Topological Mesh Irregularities:*

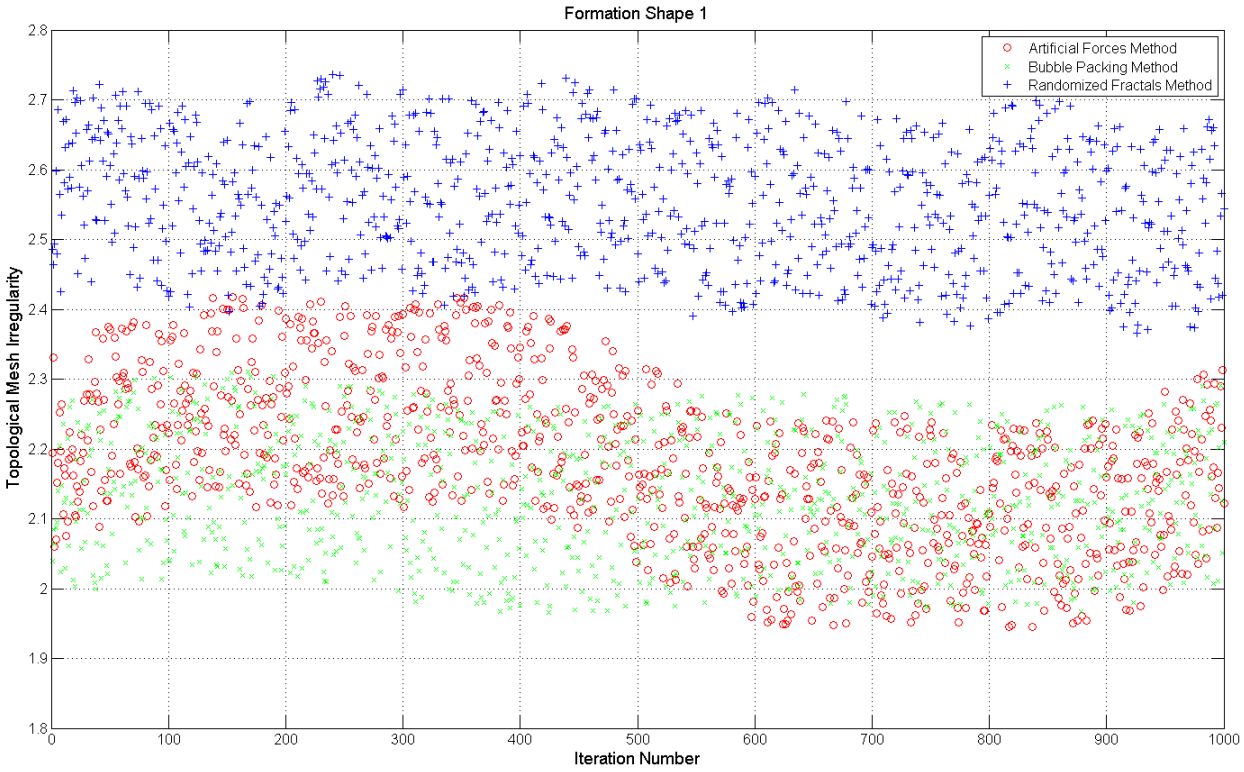


Figure 20: *Formation Shape 1 Geometrical Mesh Irregularities:*

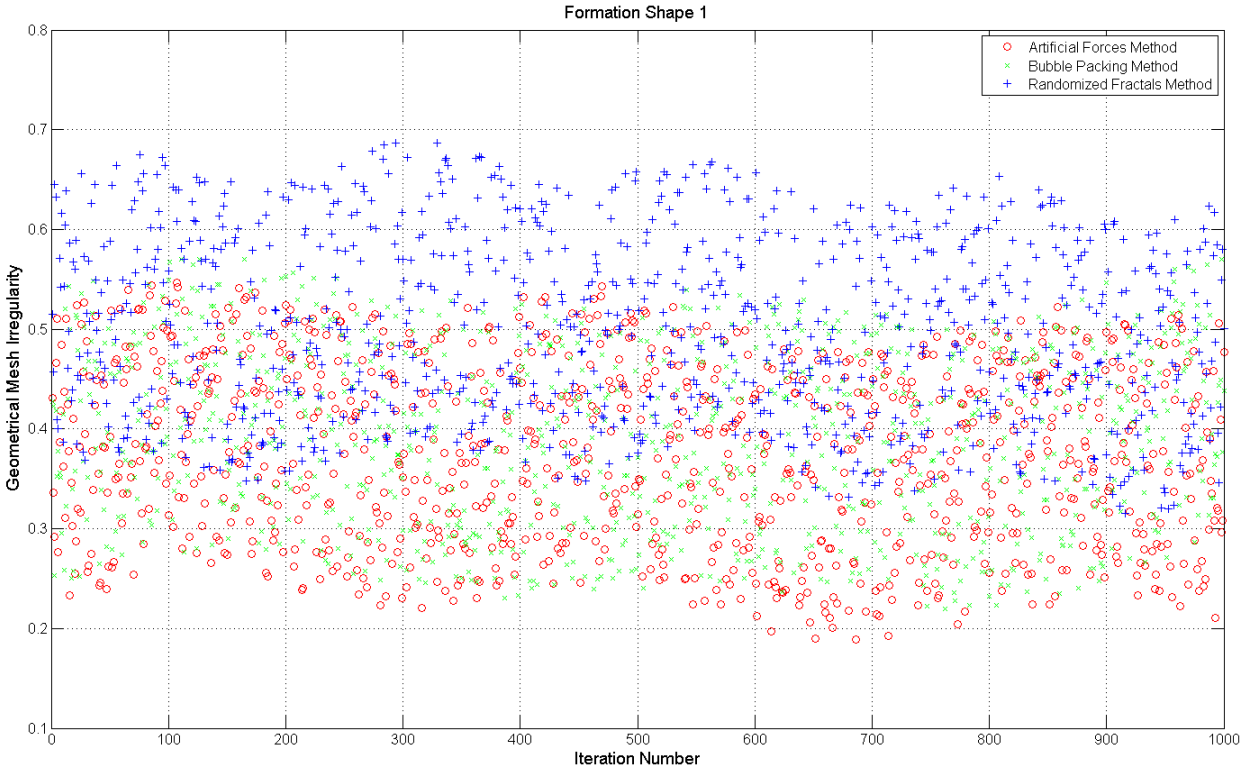


Figure 21: *Formation Shape 2 Topological Mesh Irregularities:*

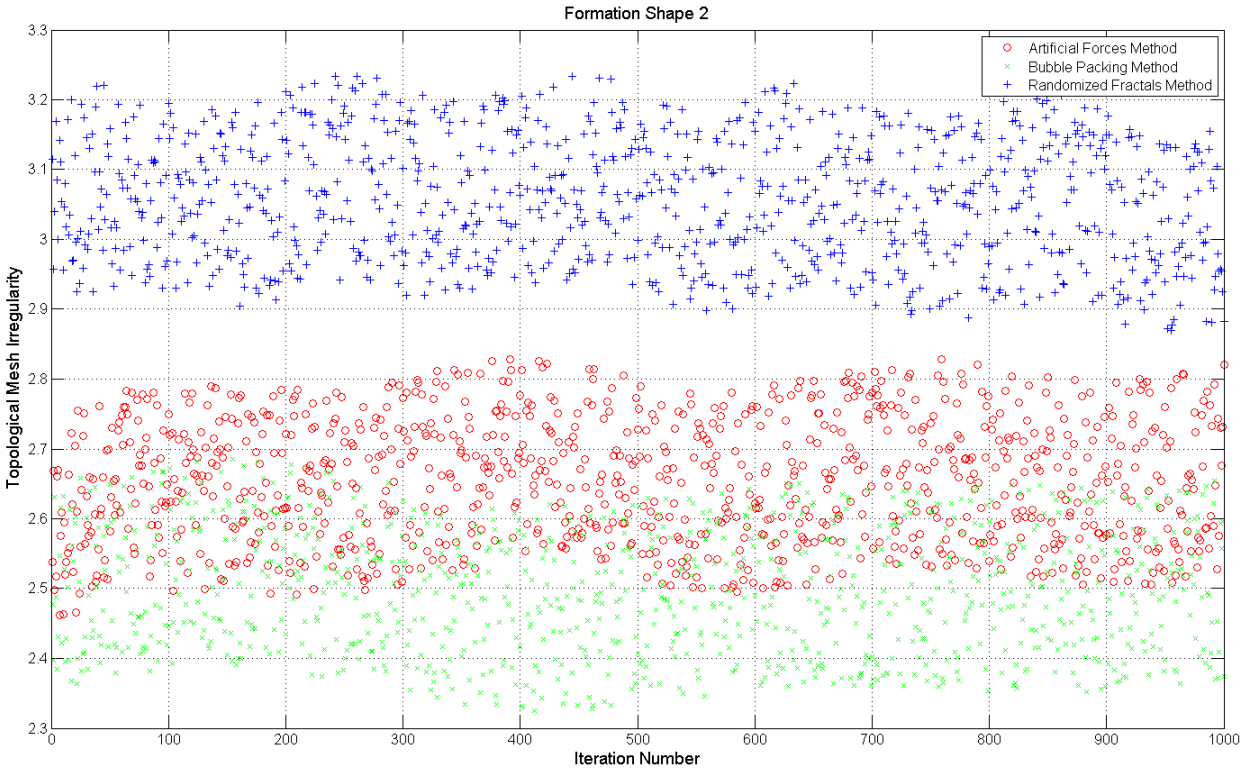
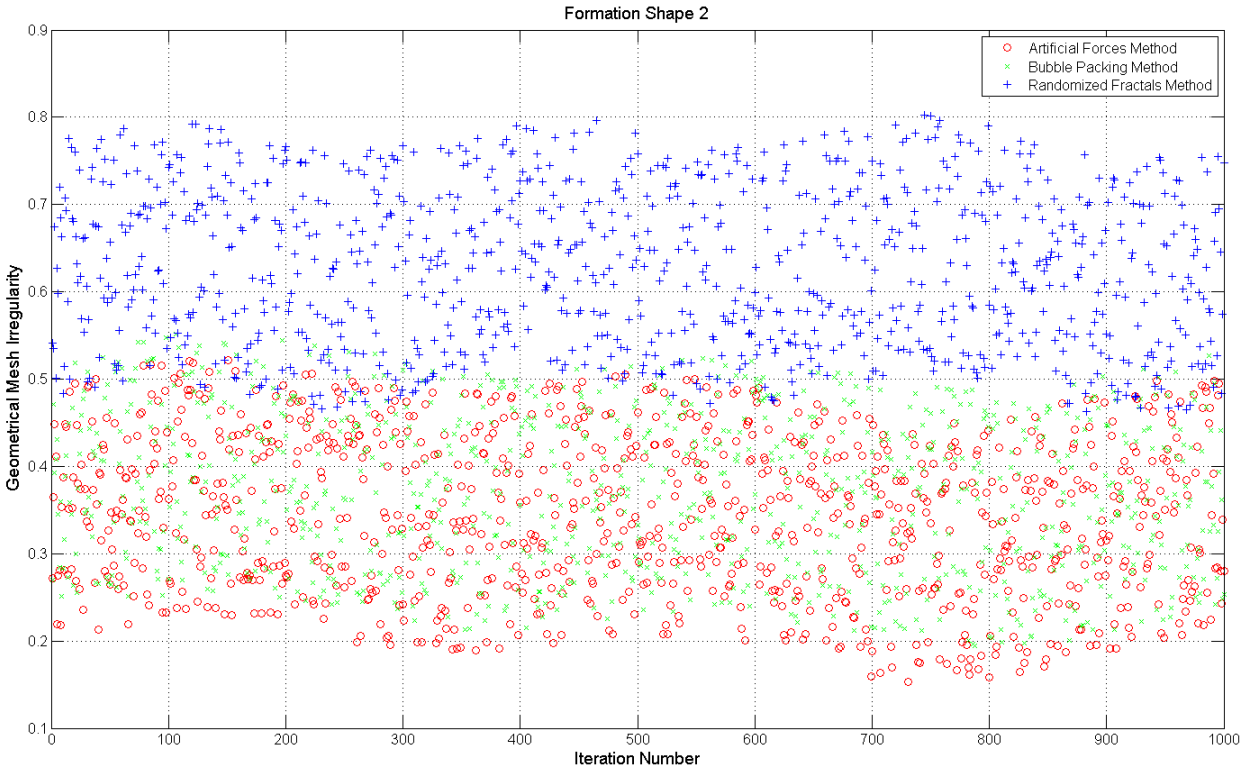


Figure 22: *Formation Shape 2 Geometrical Mesh Irregularities:*



III.2 Total Energy Consumption

In this thesis work, it is aimed to minimize the overall energy consumption of the swarm while getting the desired formation shape. Here the energy consumption of the swarm is defined as the sum of the consumed energy by the individual agents. Monte Carlo simulations with 1000 iterations are handled for the same formations shapes with different same initial conditions of the agents in the environment. Trajectories of the agents are recorded from the same initial positions until the goal states for three different types of formation control systems. Total energy consumption of the swarm is measured with total position displacements of the agents while getting the desired formation shapes. Sample outputs for three different types of formation control algorithms are illustrated in the following Figures.

III.3 Formation Shape 1

Figure 23: *Formation Shape 1 in MATLAB environment:*

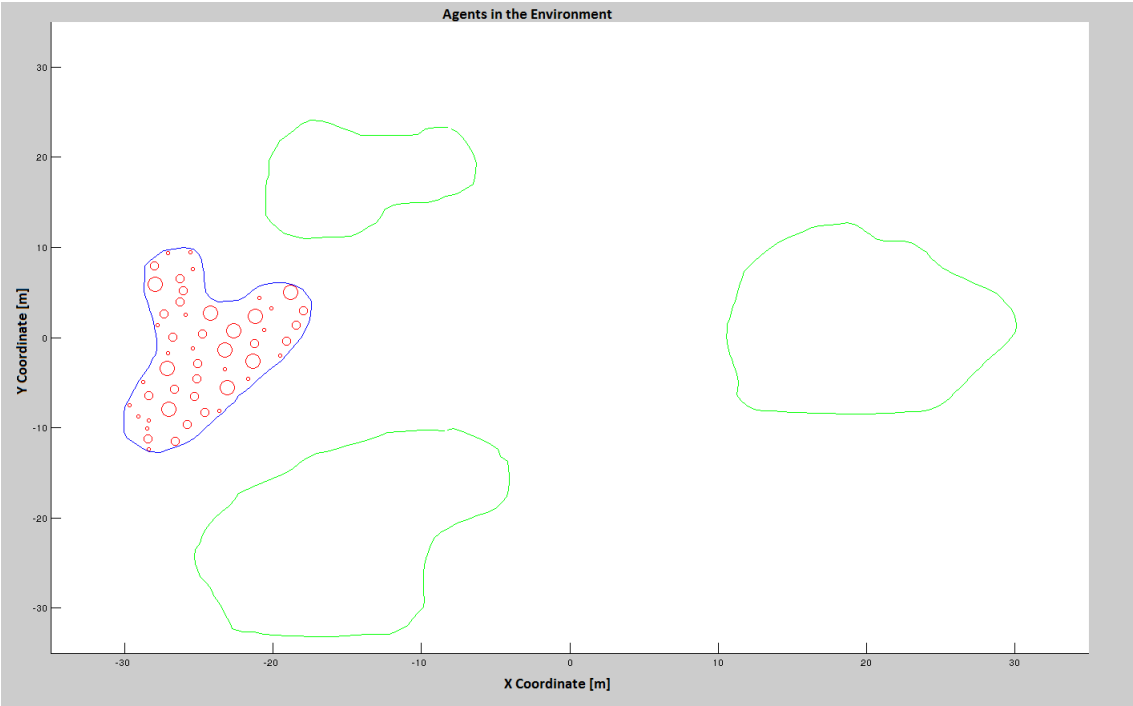


Figure 24: *Formation Shape 1 in Gazebo environment:*

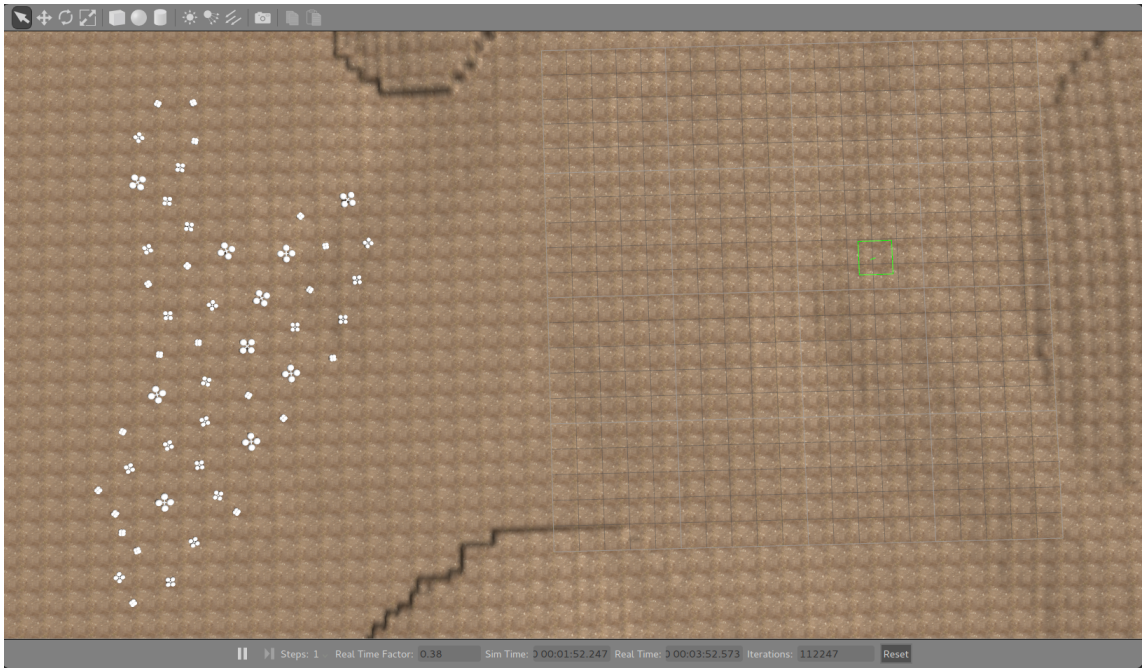


Figure 25: *Artificial Forces Method Trajectories for Shape 1*

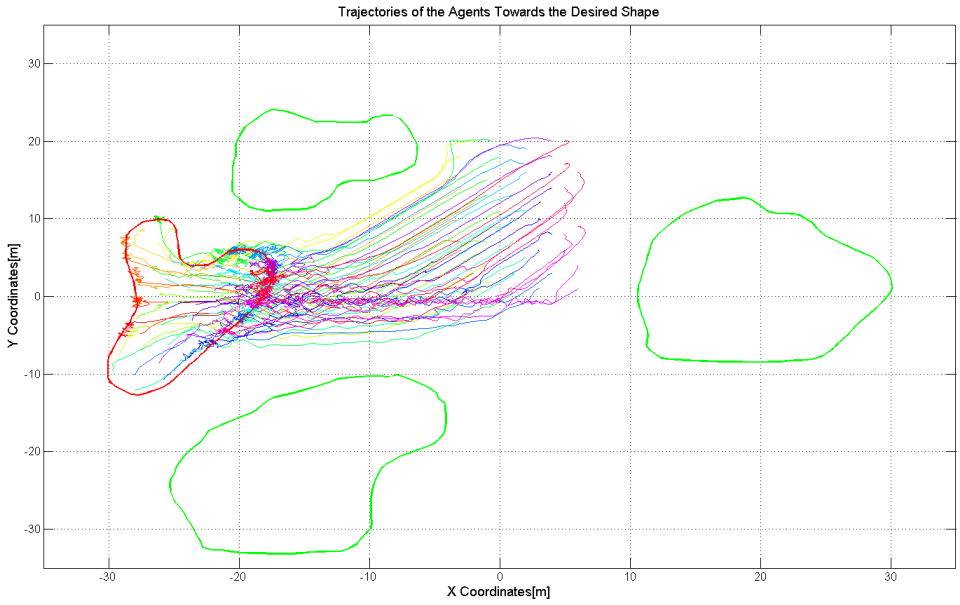


Figure 26: *Bubble Packing Method Trajectories for Shape 1*

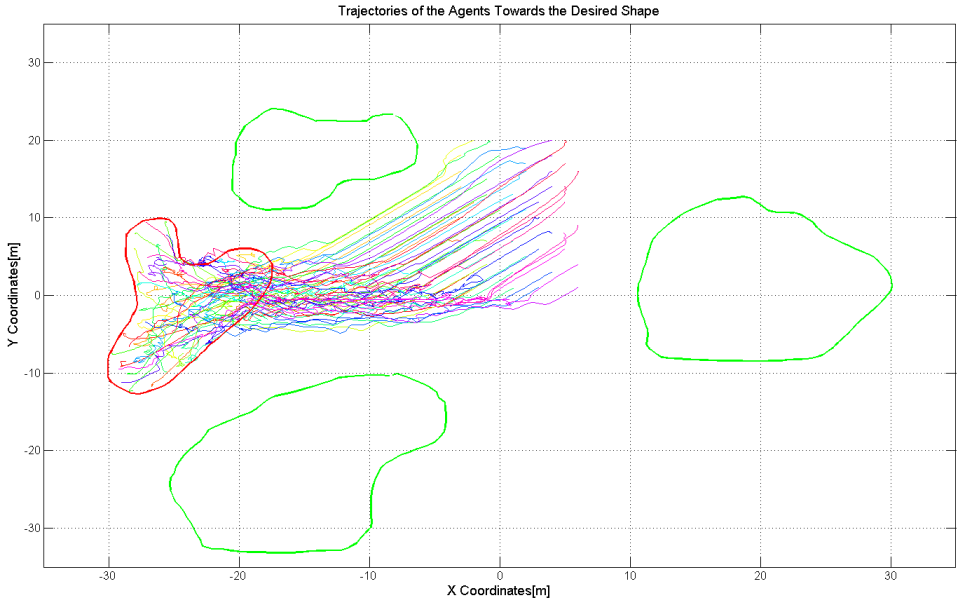
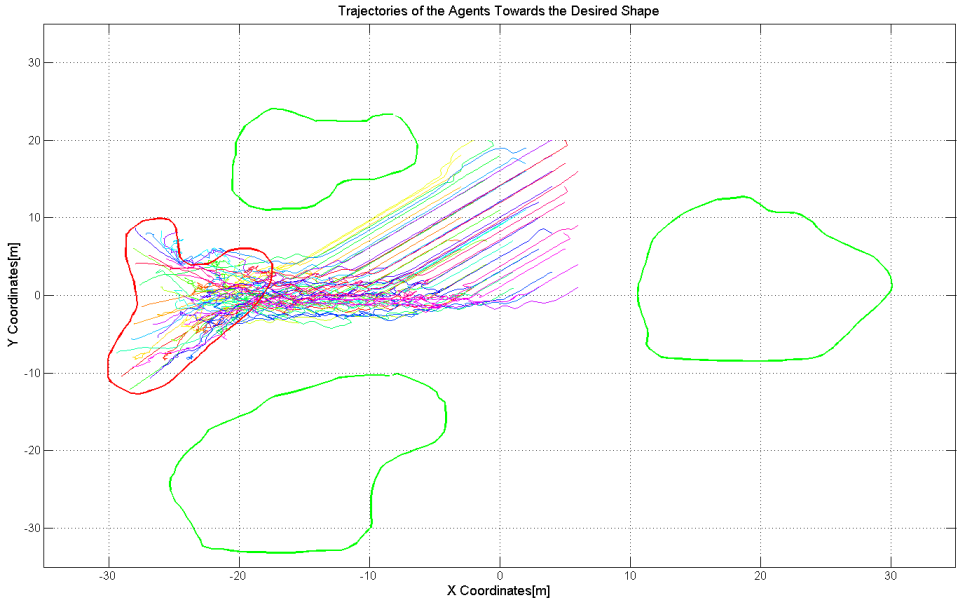


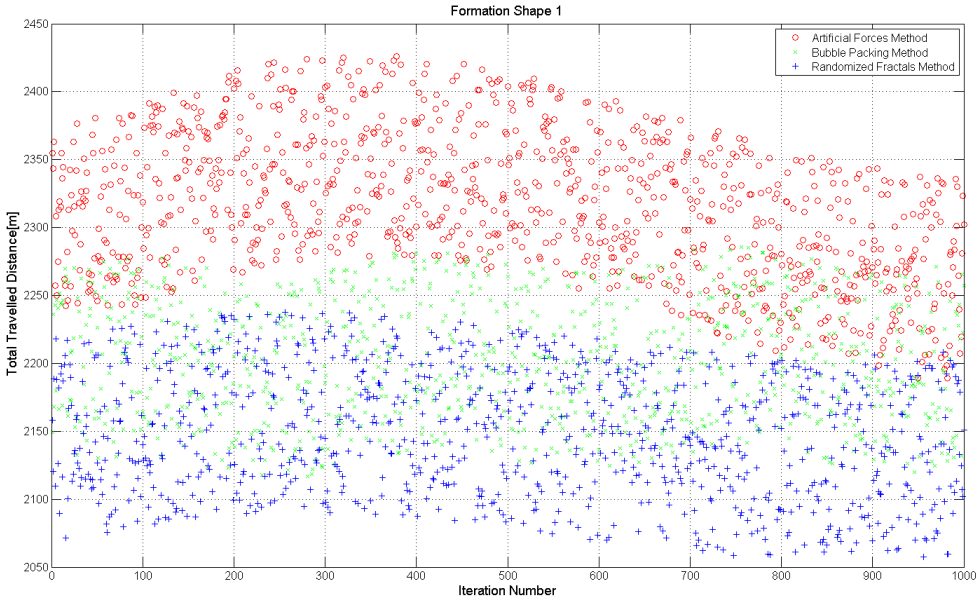
Figure 27: *Randomized Fractals Method Trajectories for Shape 1*



It is obvious that the trajectories of the agents are more chaotic and complex in the Artificial Forces method according to the other two solutions. This is because the agents are under the effect of a

total force which is instantly changing both amplitude and direction with the local interactions in the environment. According to the X-Swarm theory which is described in Section-xx, all agents will move towards the center of the formation shape if the X Swarm conditions are satisfied by the agents. So the only force component which provides the distribution of the agents in the formation shape homogenously is the intermember forces which is dynamically changing so much with the local instant neighbors of the agents in the environment. On the other hand, Bubble Packing and Randomized Fractal methods implement an algorithm in which every agent is directed to a goal state in which the total energy consumption is minimized. This approach prevents the chaotic appearance of the trajectories and minimizes the energy consumption. Monte Carlo simulations with 1000 iterations are held and the results are presented in Figure -xx.

Figure 28: *Total Energy Consumption*



III.4 Formation Shape 2

Second formation shape gives similar results with three different formation control methods, the Artificial Forces Method increases the total energy consumption of the agents because of the reasons discussed for the Formation Shape 1.

Figure 29: *Formation Shape 2 in MATLAB environment:*

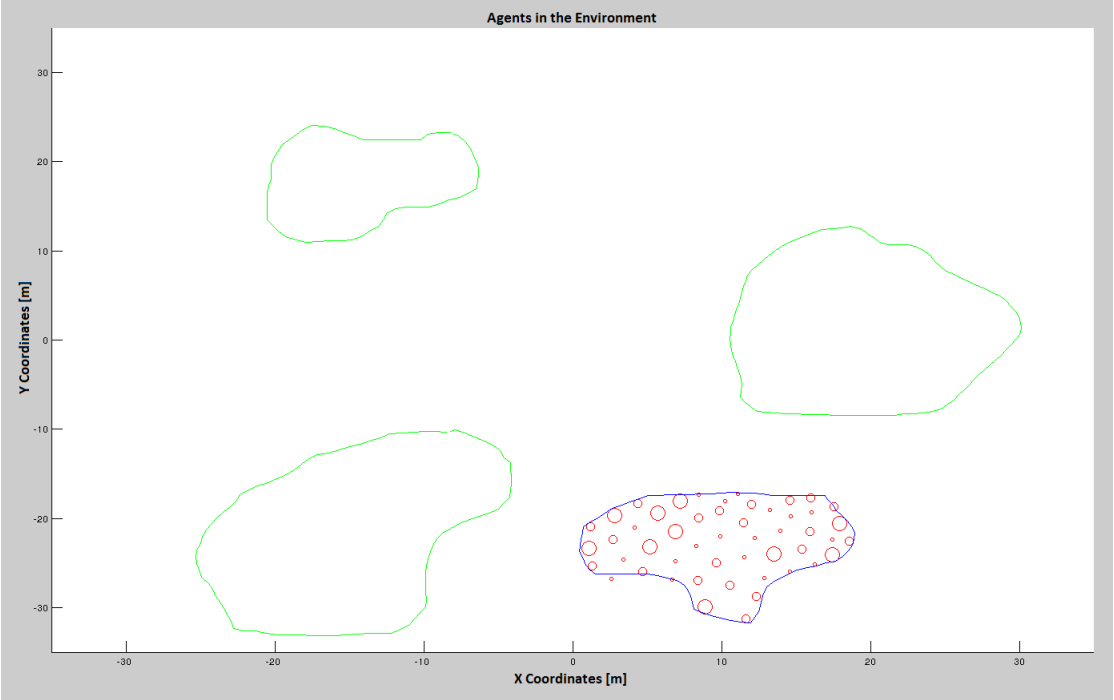


Figure 30: *Formation Shape 2 in Gazebo environment:*

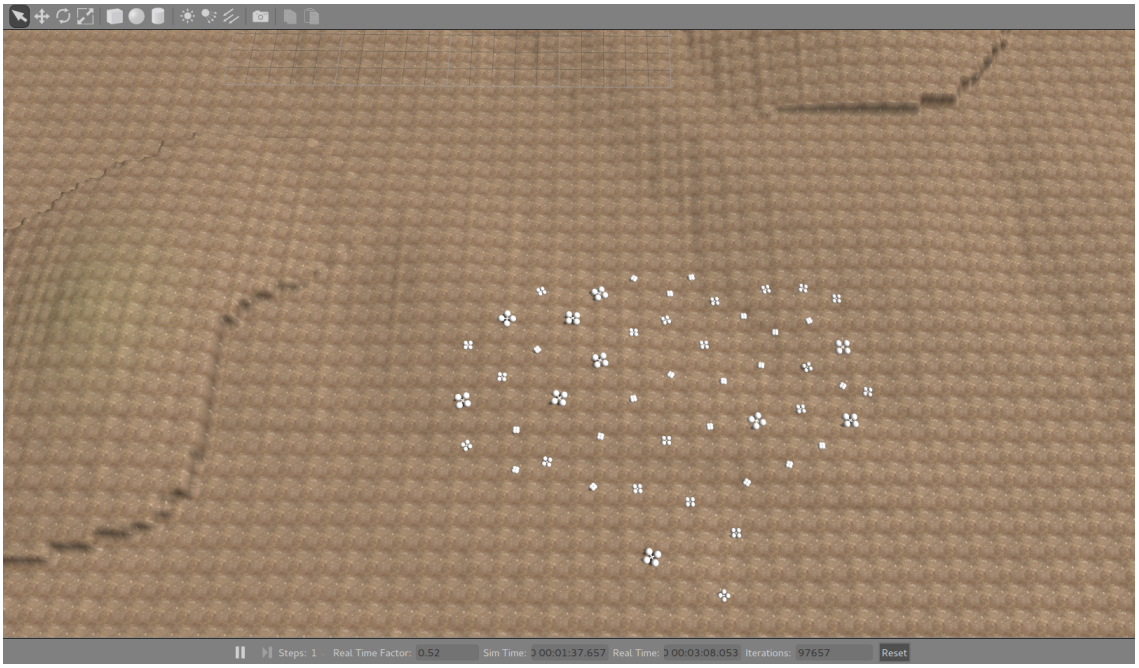


Figure 31: *Artificial Forces Method Trajectories for Shape 2*

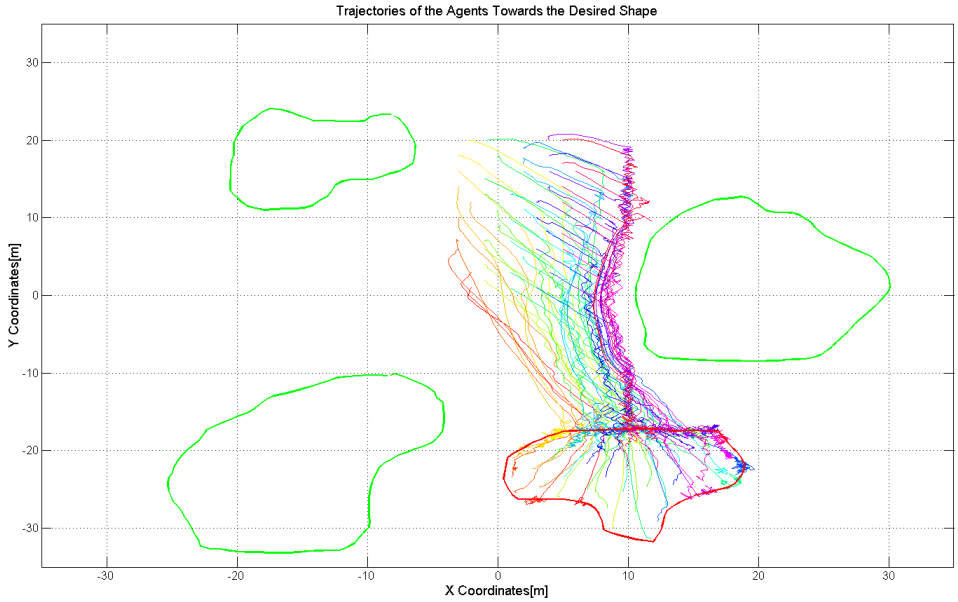


Figure 32: *Bubble Packing Method Trajectories for Shape 2*

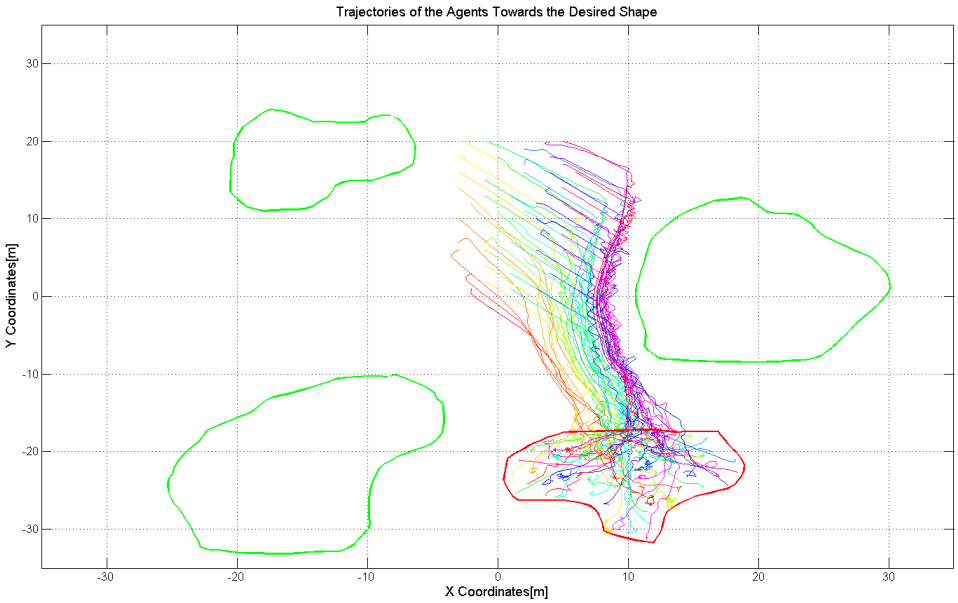
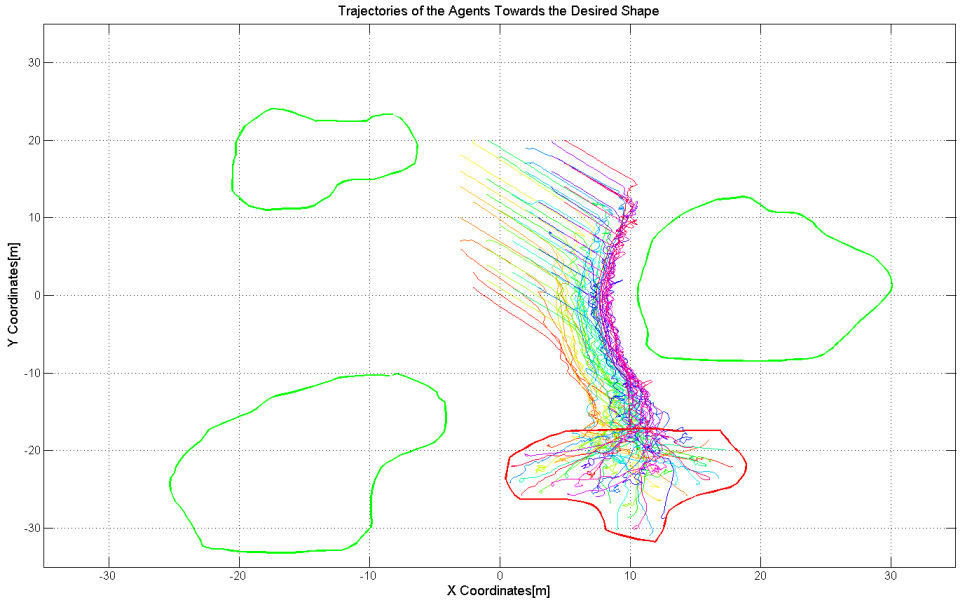
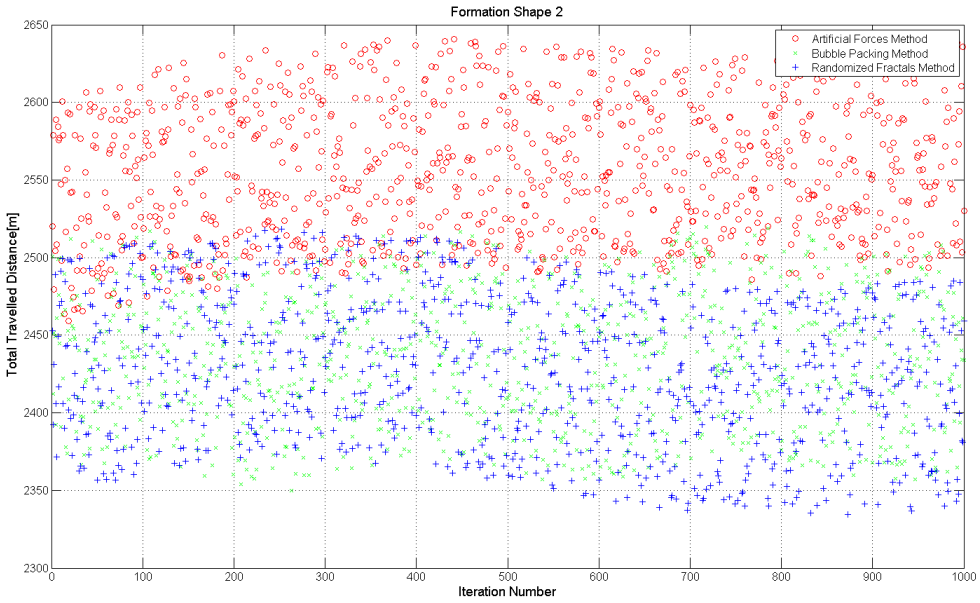


Figure 33: *Randomized Fractals Method Trajectories for Shape 2*



The results of the Monte Carlo simulations with 1000 iterations are illustrated in Figure -xx /

Figure 34: *Total Energy Consumption*



III.5 Settling Time

Settling time is defined as delta time "$t_{final} - t_{start}$" ,where t_{start} is the initial time and t_{final} is the time that the all of the agents are inside of the desired formation shape and the norm of the velocity vector for each agent $\|v_i\|$ is

$$\|v_i(t)\| < 0.01\ [m/sec] \quad \forall t > t_{final} \tag{110}$$

Settling times are measured for the simulations held in Energy Consumption Part-xx, and Artificial Forces method have the worst performance as expected. Since there are no predetermined goal states for the agents in the desired formation shape, the settling of the agents in random places under the equilibrium of the artificial force components takes much more time than the other two formation control methods. The agents have reached the steady state at their goal states with the shape partitioning methods faster than the Artificial Forces Methods. The results are illustrated in Figure-xx.

Figure 35: *Total Settling Time for Shape 1*

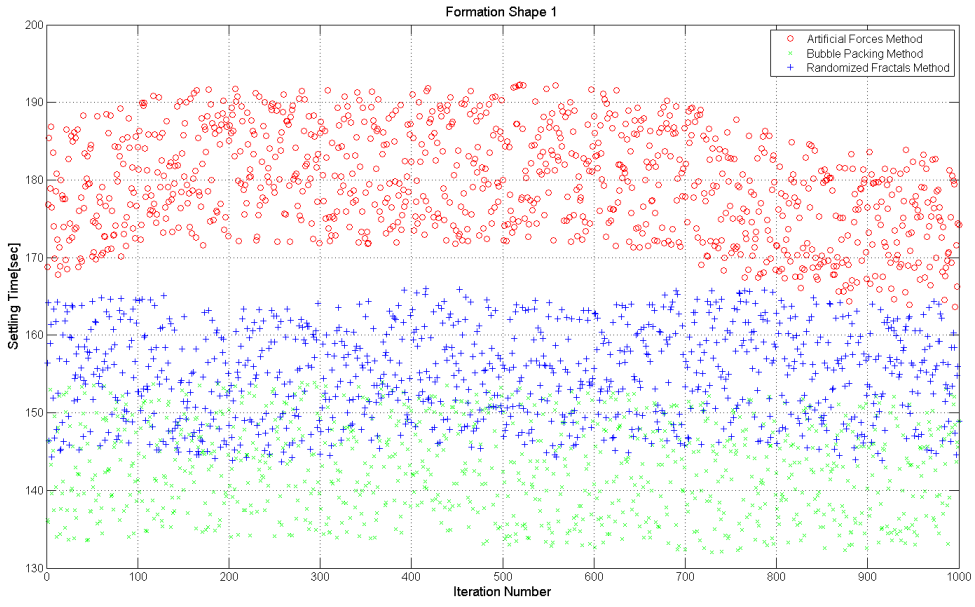
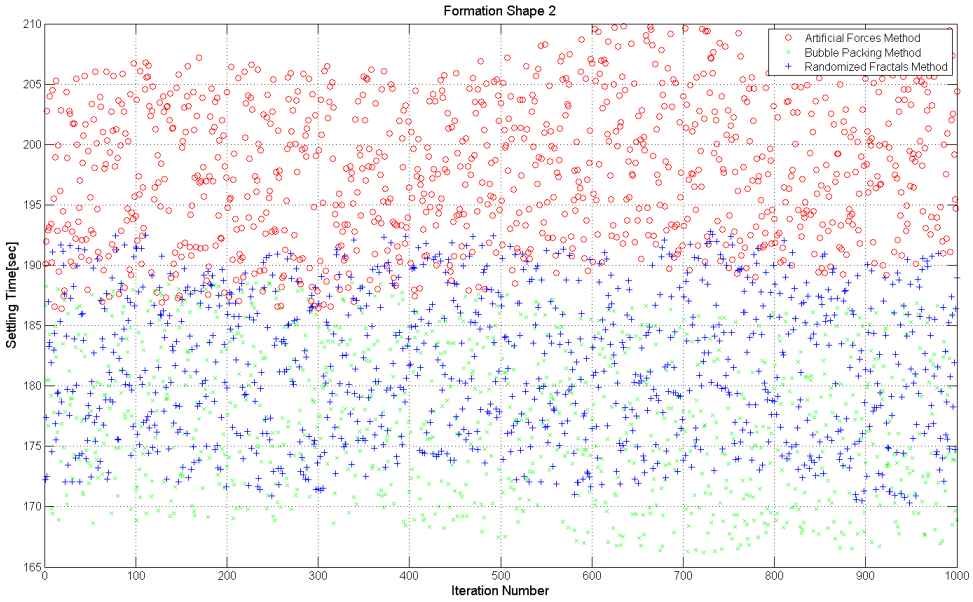


Figure 36: *Total Settling Time for Shape 2*



III.6 Evaluation

Three different formation control approaches are evaluated with different kinds of metrics including the total energy consumption, settling time and mesh quality which is a measure of how the agents homogenously distributed while covering the desired formation shape. Artificial forces method have the worst performance in setlling time and energy consumption metrics due to the absence of predetermined goal states which is discussed in Section-xx in details. On the other hand the Randomized Fractals method have the worst mesh quality performance because of its randomized nature of assignment the goal states in the desired formation shape. The methods which have the worst performance at the related metrics are illustrated in Table -xx

Table 1: *Formation Control Methods with Worst Performance*

Method/Metric	Energy Consumption	Settling Time	Mesh Quality
Artificial Force	X	X	
Bubble Packing			
Randomized Fractals			X

It is obvious that time Bubble Packing Method have the best performance for all metrics. In fact, Bubble Packing method combines the efficient approaches of the other two methods. In the shape partitioning phase of the algorithm, it uses interbubble forces which have a similar structure to the intermember forces applied by the Artificial Forces method in real time. This force have the greatest effect on the homogeneity of the agents in the desired formation shape because it provides a global consensus for the agents in which each agent reaches an equilibrium state under the total forces acting by their neighbors and the formation shape. On the other hand, bubble packing method implements an algorithm in which every agent assigned to a goal state instantly at each execution time of the algorithm to minimize the overall energy consumption o the swarm.

This approach reduce the settling time and the total displacements of the agents from the initial state to their final states since their final positions in the formation shape is predetermined. It is possible that the agents' goal states may be changed with the execution of the algorithm at different steps, but these assignments will converge to a uniqe subset while the agents are getting closer to the goal states.

Some other formation shape trials with the Bubble Packing algorithm is illustrated in the follwoing Figures-xx.

Figure 37: *Formation Shape - 1 in Gazebo Environment*

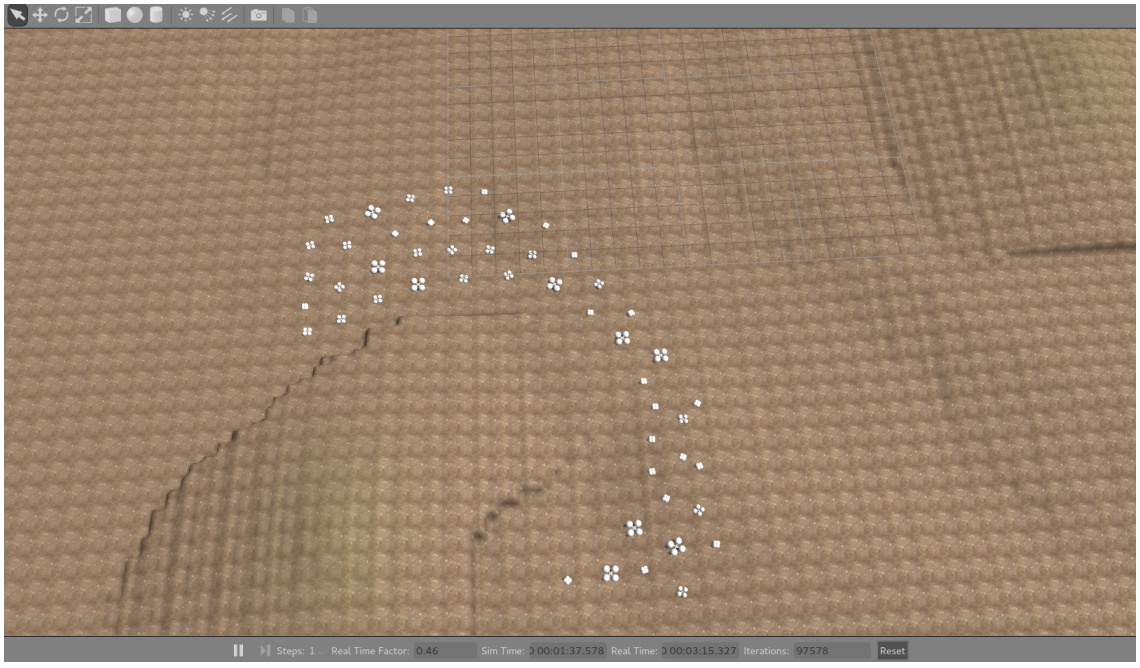


Figure 38: *Formation Shape - 1 in MATLAB Environment*

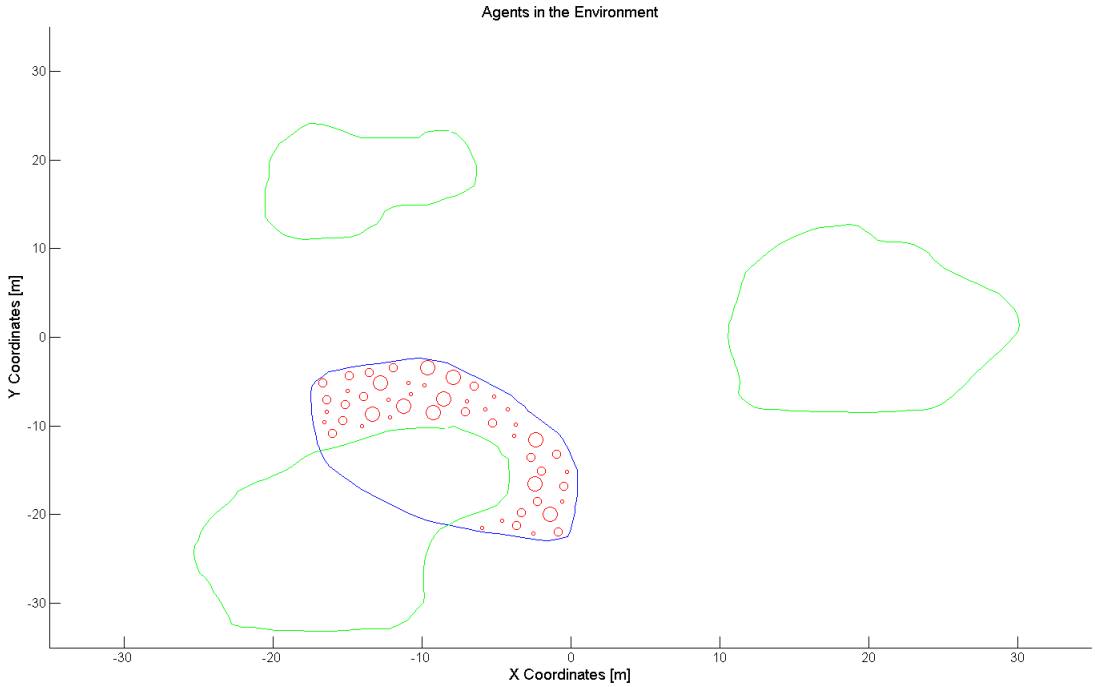


Table 2: *Performance Metrics for Shape - 1*

Energy Consumption[m]	Settling Time[sec]	Topological Mesh Irregularity	Geometrical Mesh Irregularity
1889	164	2.54	0.62

Figure 39: *Formation Shape - 2 in Gazebo Environment*

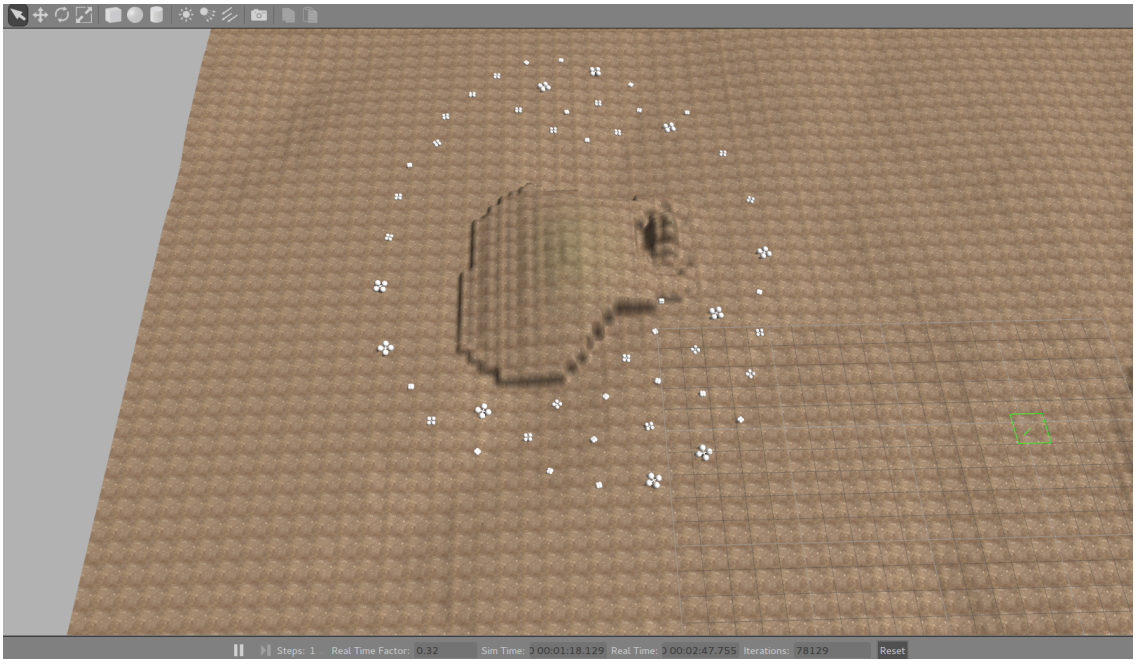


Figure 40: *Formation Shape - 2 in MATLAB Environment*

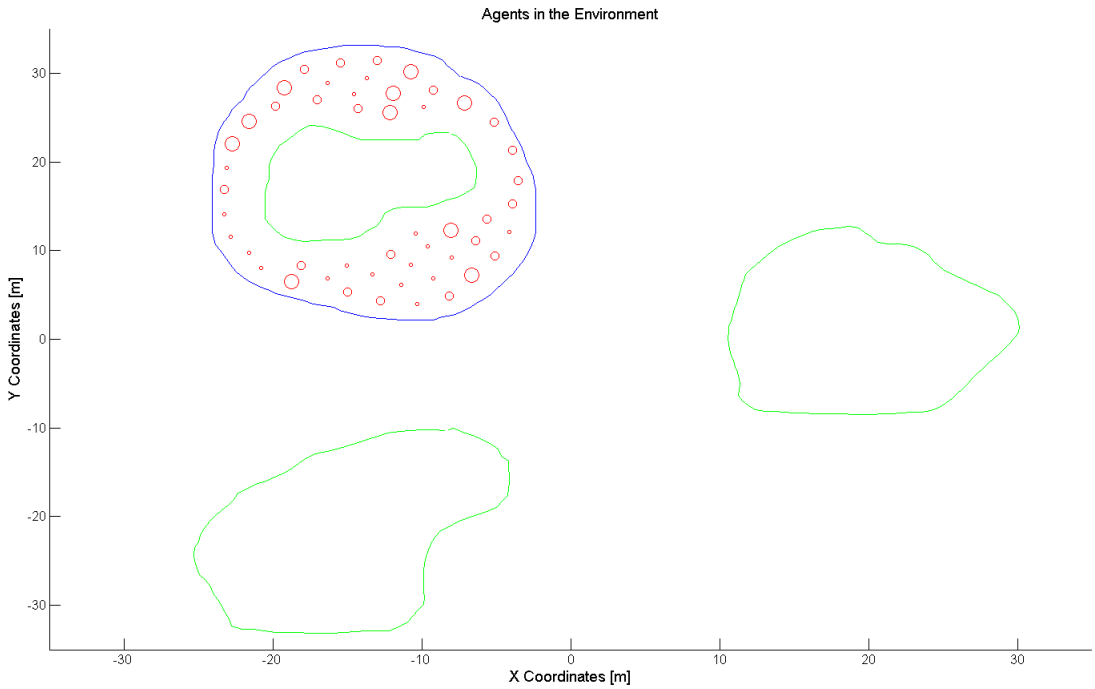


Table 3: *Performance Metrics for Shape - 2*

Energy Consumption[m]	Settling Time[sec]	Topological Mesh Irregularity	Geometrical Mesh Irregularity
1611	153	2.95	0.73

Figure 41: *Formation Shape - 3 in Gazebo Environment*

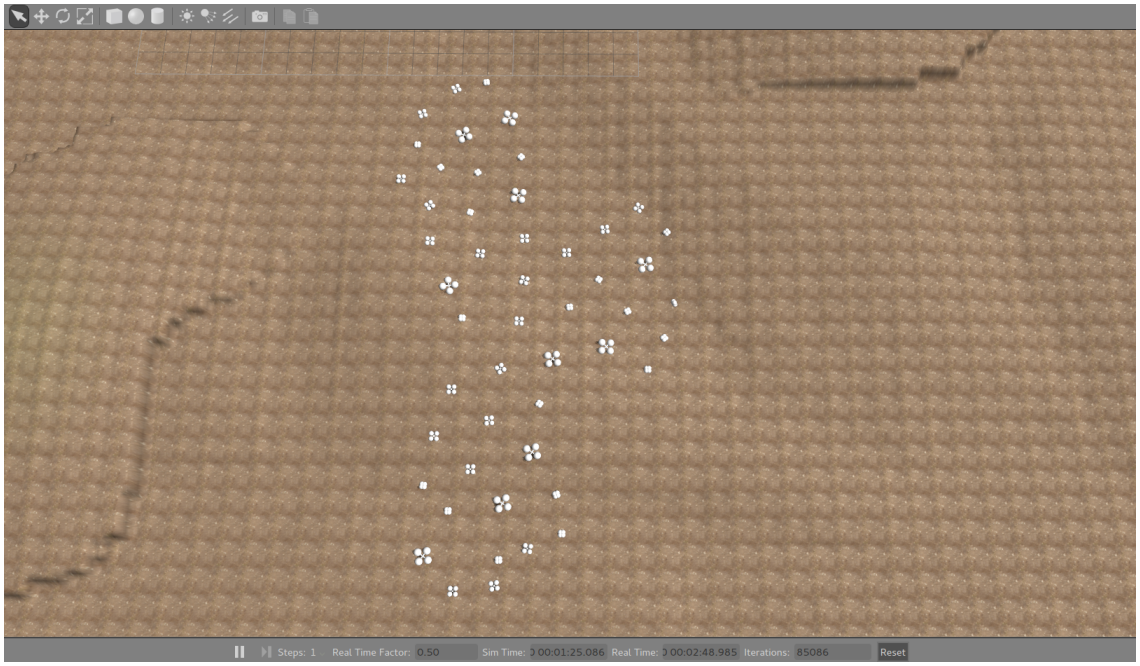


Figure 42: *Formation Shape - 3 in MATLAB Environment*

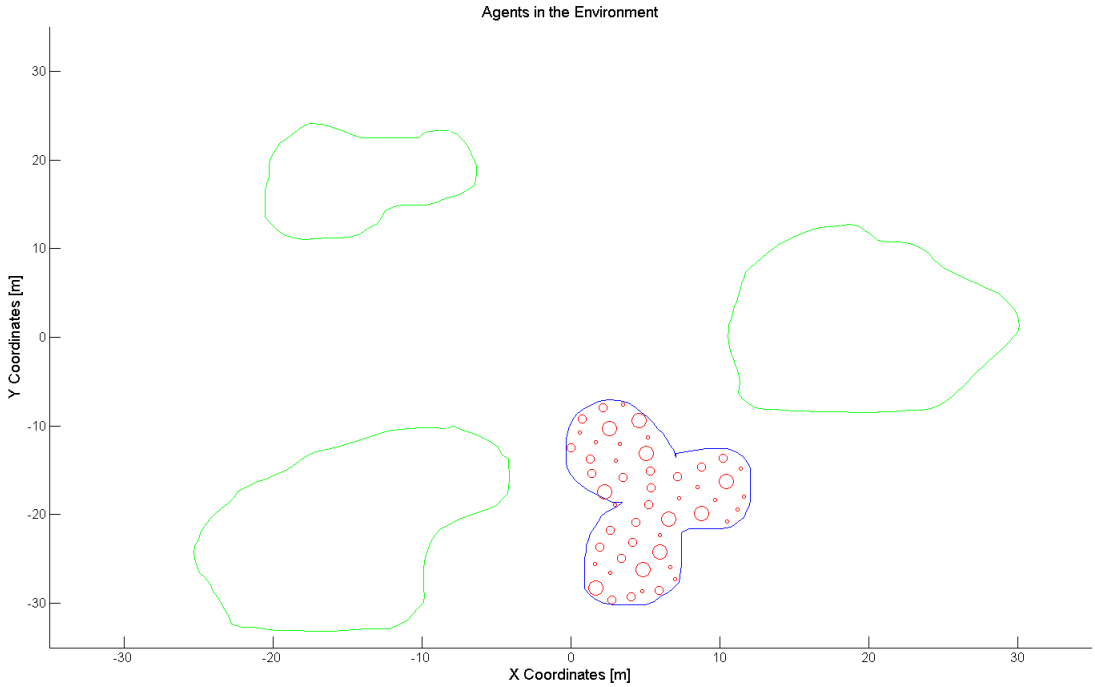


Table 4: *Performance Metrics for Shape - 3*

Energy Consumption[m]	Settling Time[sec]	Topological Mesh Irregularity	Geometrical Mesh Irregularity
1432	126	2.23	0.45

Figure 43: *Formation Shape - 4 in Gazebo Environment*

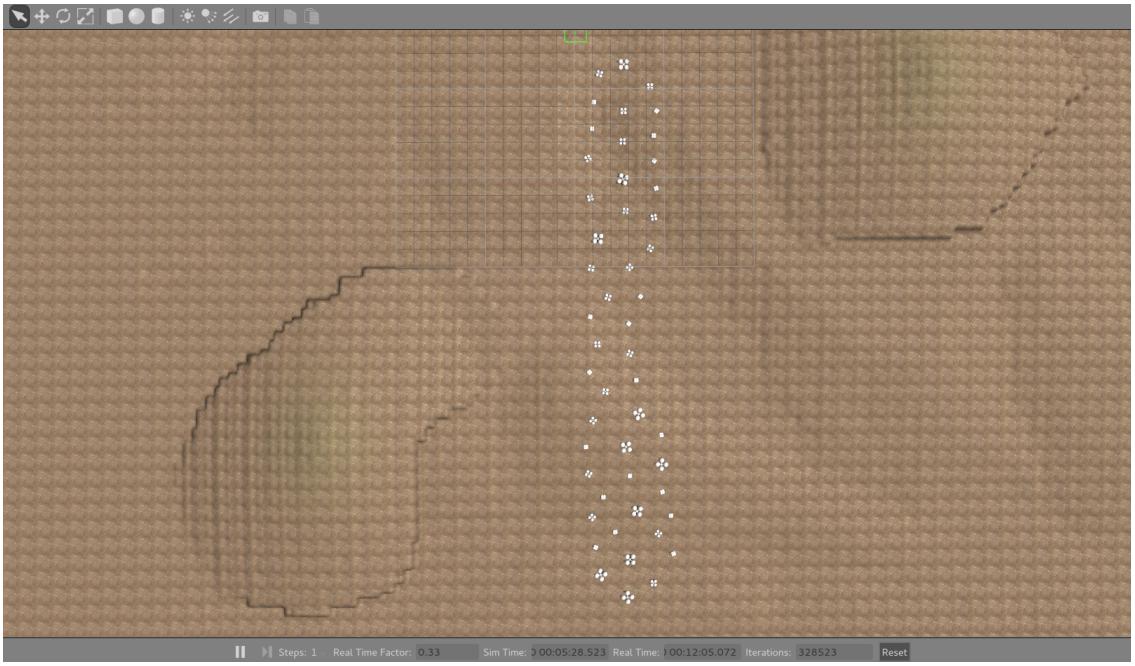


Figure 44: *Formation Shape - 4 in MATLAB Environment*

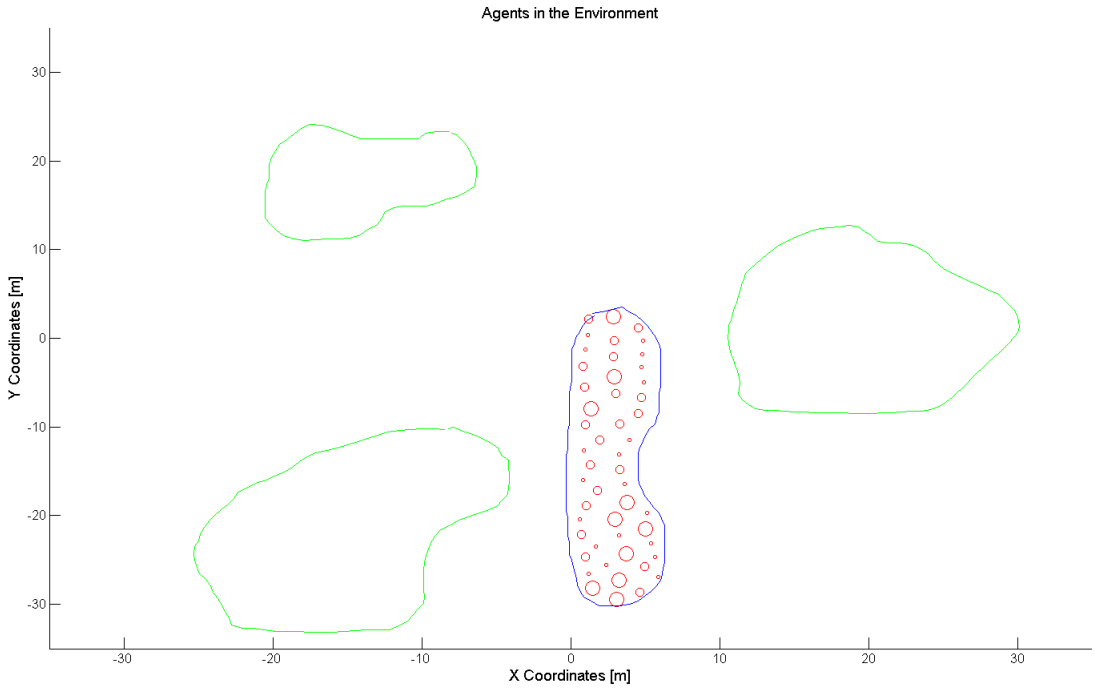


Table 5: *Performance Metrics for Shape - 4*

Energy Consumption[m]	Settling Time[sec]	Topological Mesh Irregularity	Geometrical Mesh Irregularity
1578	142	2.15	0.56

A simulation with multiple formation shapes sequentially is illustrated in Figure-xx.

Figure 45: *Multiple Formations*

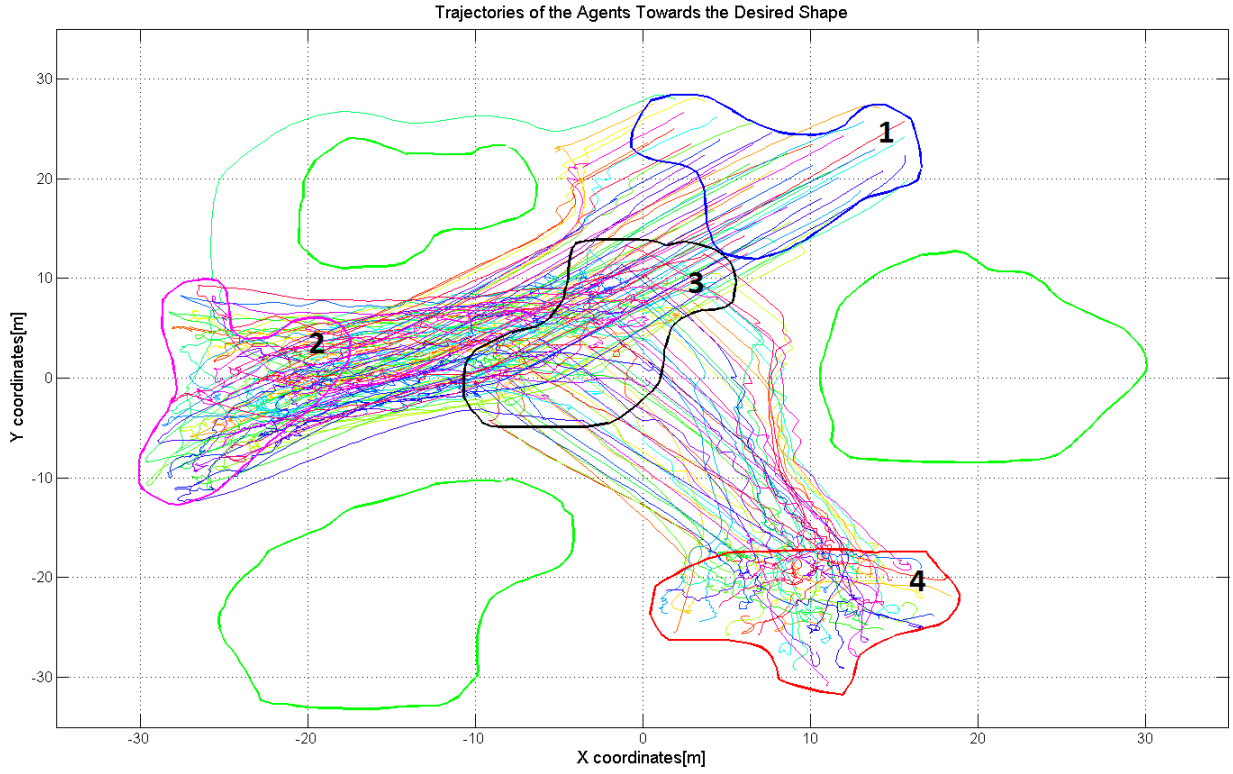


Table 6: *Performance Metrics for Multiple Formation Shapes*

Total Energy Consumption[m]	Total Settling Time[sec]	Topological Mesh Irregularity Mean	Geometrical Mesh Irregularity Mean
4216	435	2.21	0.68

IV. HARDWARE IMPLEMENTATION

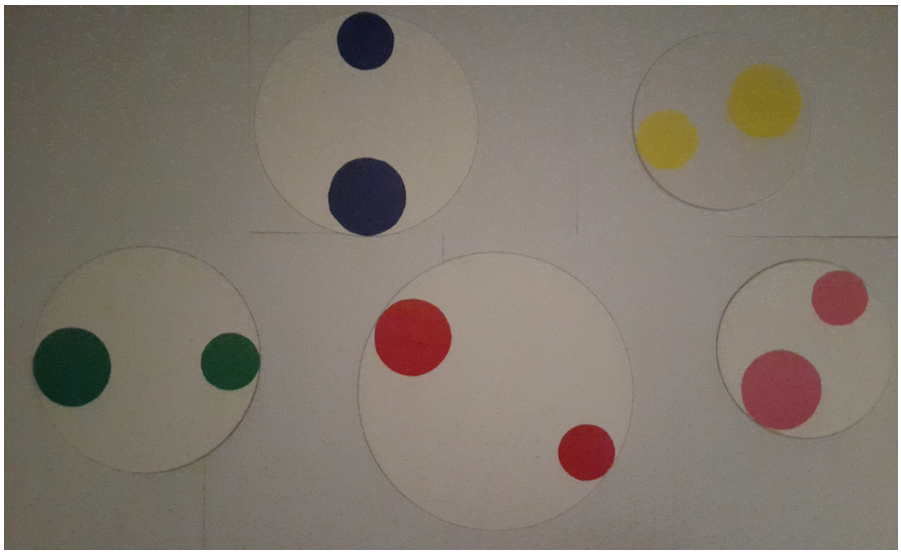
A hardware demonstration is held with 5 mobile robots with different sizes, to achieve a proof of concept related with the topics searched in this thesis work. The agents are designed as mobile robots with three omni-wheels which allows them to navigate in the field in every direction without the need for changing their attitudes (i.e. headings) . The schematic environment of this implementation is illustrated in Figure -xx

Hardware Implementation semasi sunumdan buraya eklenecek.

Each agent and the mission computer has a radio link to create a mesh network in which every agent and the mission computer can transfer messages between each other directly or with the help of their neighbors. Also they can broadcast messages to the rest of the nodes in the network. On the other hand each agent has their individual CPU to execute the goal state decision process and to control the actuators to achieve the desired goal states. These processors also executes the control system which is descired in Section -xx and they manage the messaging by implementing the related protocols predetermined. This architecture supports the idea of partially decentralized formation control in which each agent is responsible to take decisions and reach to a global consensus with the rest of the swarm on the potential goal states. These potential goal states are determined by the mission computer which takes the desired formation shape from the operator and executes shape partitioning algorithms. The data of the potential goal states is broadcasted to all of the agents in the environment with the help of mesh network.

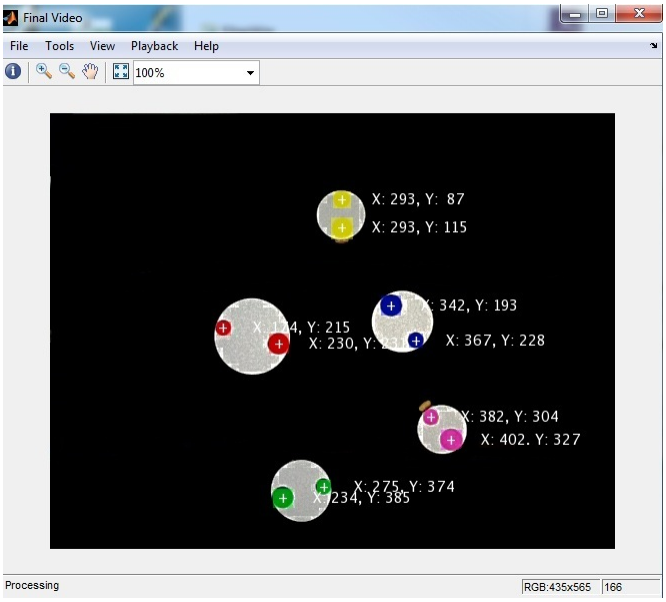
Since the environment is an indoor area, it is impossible to use a GNSS system to provide position and velocity measurements to the agents. To emulate this external measurements a visual feedback system is used with the help of a E/O camera and image processing algorithms. The image processing algorithms used in this work depends on the color classfication of different cover planes placed on the top of the mobile robots. These covers are illustrated in Figure -xx

Figure 46: *Covers for Different Types of Agents*



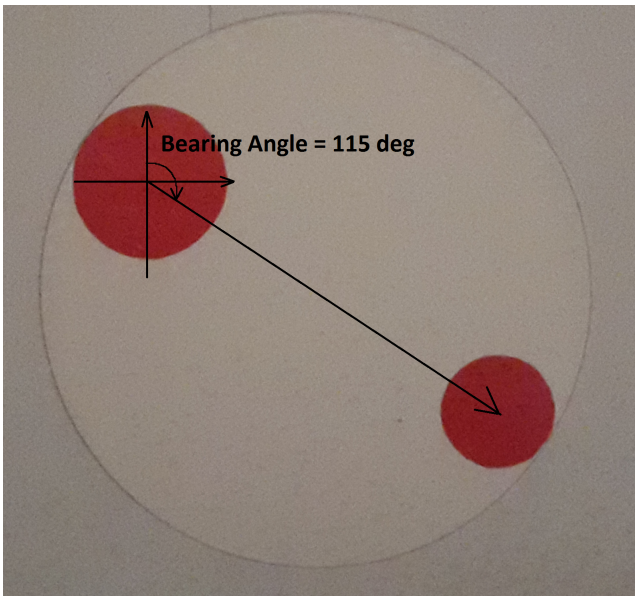
Covers have different sizes which represents the volumes of the agents in 2D environment. Each cover have two different size of circles with the same color. The colors of the circles are used to classify the agent ids and the positions of the circles are used to determine both position and attitude (i.e. heading) angles of the mobile robots in the environment. The orientation data of the agent in the environment is used in the command mixing algorithms of omni - wheels which will be described in details. The video of the environment is transmitted to the mission computer in nearly real time. Mission computer executes the image processing algorithms which filters the desired colors and detects the positions of the colored circles and broadcasts the position and orientation datas of the agents. A sample output of the image processing algorithm is illustrated in Figure-xx.

Figure 47: *Sample Output of the Image Processing Algorithm*



The orientation of an agent is determined with the clockwise bearing angle of the vector from the center of the large circle to the center of the small circle. Figure-xx illustrates this calculation.

Figure 48: *Orientation of an Agent in the Environment*



Each agent have their individual processor unit and radio link on their boards. A block diagram of an individual agent is illustrated in Figure -xx

Sunumdan Single agent sekli konacak

Microchip's Pic16f690 Microcontroller is responsible of controlling the mobile robot and

data communication with the environment. This microcontroller runs the control algorithms which is described in Section-xx and drives the 3 step motor control units, ULN2003APG, via GPIO peripherals. The instant velocity setpoints for the stepper motors are determined with the command mixture algorihtm illustrated in Figure -xx.

Sunumdan command mixture sekli eklenecek

The desired velocity vector for the agent determined by the control algorithms is distributed to the stepper motors in accordance with the orientation of the agent . Let $\|Vel\|$ be the amplitude of the desired velocity and α is the angle representing the desired direction of the movement with respect to mobile robot's body frame. The velocities for the stepper motors is determined with following equations,

$$V_A = \|Vel\| \, cos(150 + \alpha)$$
$$V_B = \|Vel\| \, cos(30' + \alpha)$$
$$V_C = \|Vel\| \, cos(270 + \alpha)$$

where V_A, V_B, V_C represents the desired velocities of stepper motor A,B and C respectively.

Microcontroller is also drives the radio link via UART peripheral and manages the communication of the agent with the environment. All of the units on the board are supplied with a 5VDC regulator, 7805. The schematic of the circuit which controls the agent is illustrated in Figure-xx.

Figure 49: *Schematic of the Circuit on the Board*

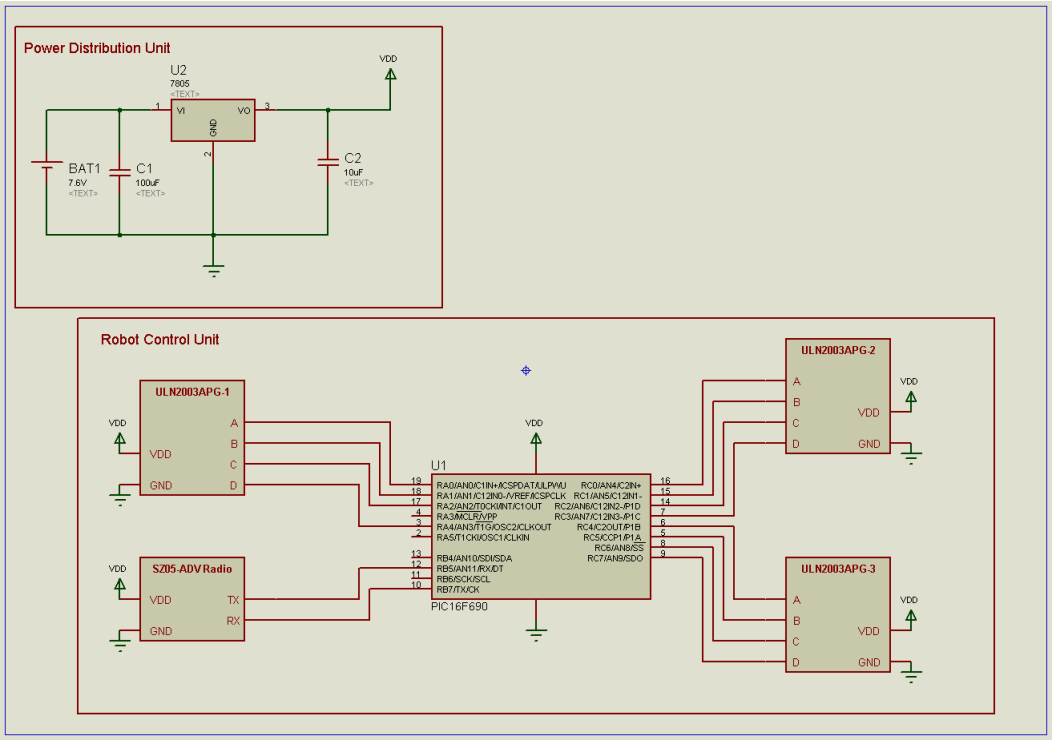


Figure 50: *3D Visualization of the Layout*

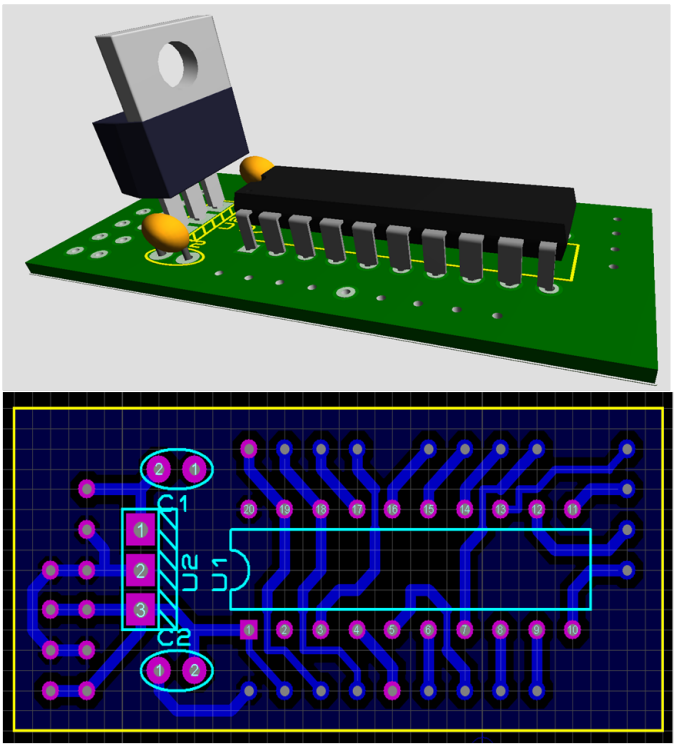


Figure-xx and Figure-xx describes the hardware parts used on a mobile robot.

Sunumdan Hardware-1 ve Hardware-2 sekilleri konacak

V. Performance Analysis

The system is tested with several formation shapes and results are analyzed with total displacements of the agents and the settling time metrics. Sample formation shapes are covered and the algorithm proposed for formation control is tested in real time successfully. Traces of the agents are plotted with the continuous blue lines from their initial positions to the goal states in the following figures. The desired formation shape is plotted with black circles.

Figure 51: *Formation Shape 1- Matlab Environment*

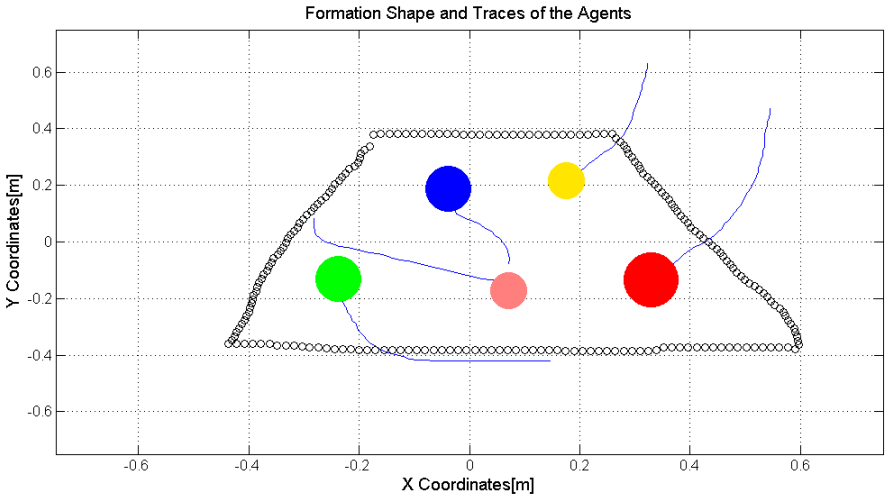


Figure 52: *Formation Shape 1- Test Environment*

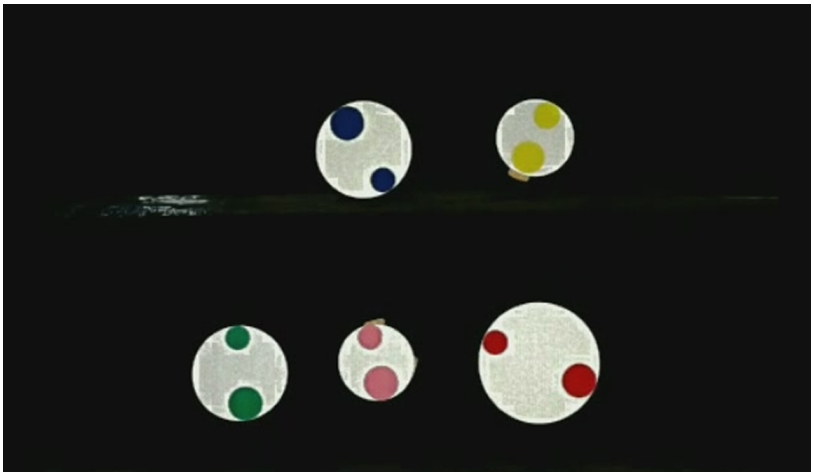


Table 7: *Performance Metrics for Shape - 1*

Total Displacements[m]	Settling Time[sec]
1.75	23

Figure 53: *Formation Shape 2- Matlab Environment*

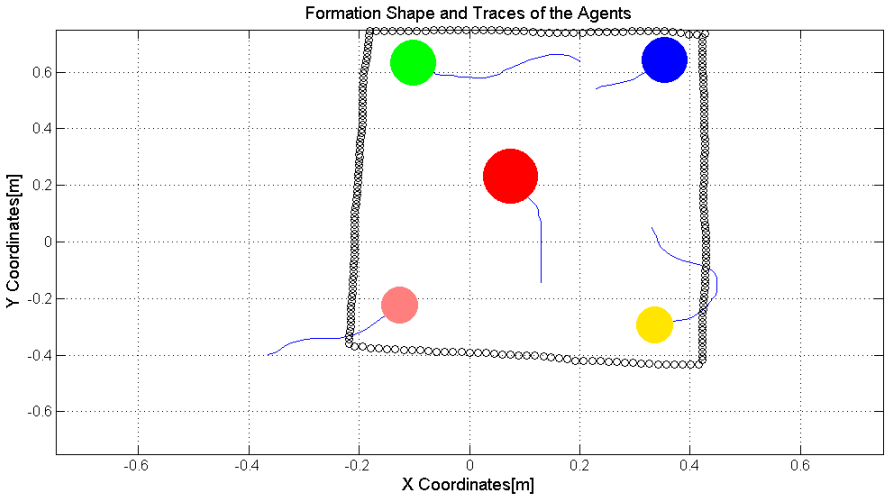


Figure 54: *Formation Shape 2- Test Environment*

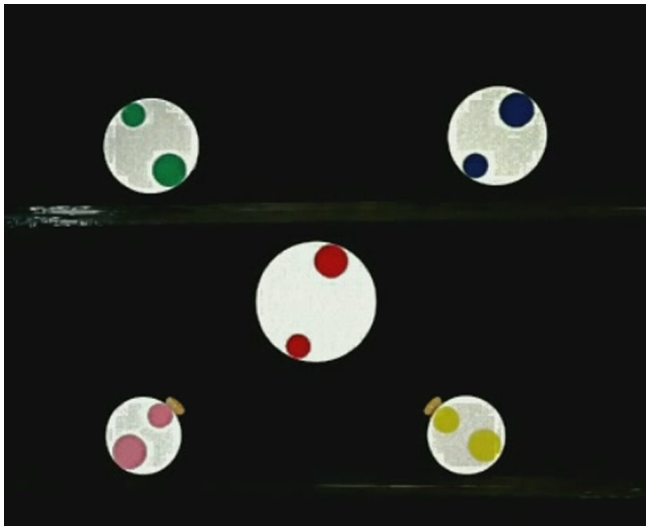


Table 8: *Performance Metrics for Shape - 2*

Total Displacements[m]	Settling Time[sec]
1.63	19

Figure 55: *Formation Shape 3- Matlab Environment*

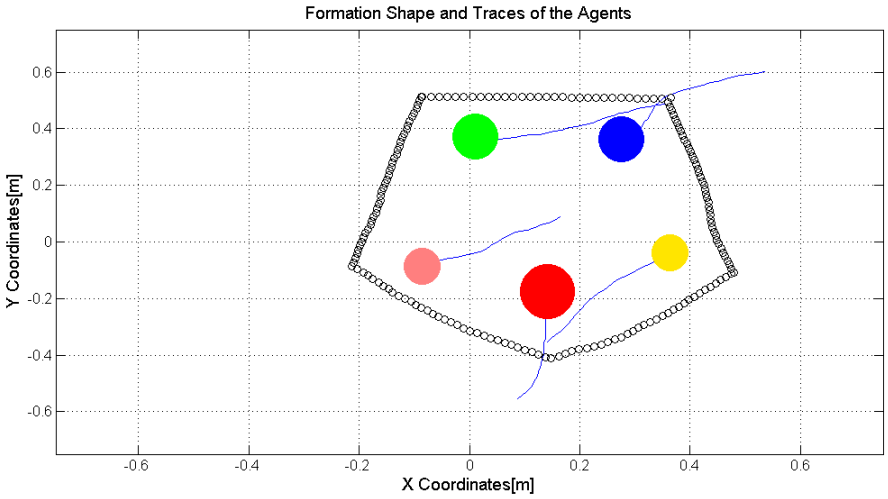


Figure 56: *Formation Shape 3- Test Environment*

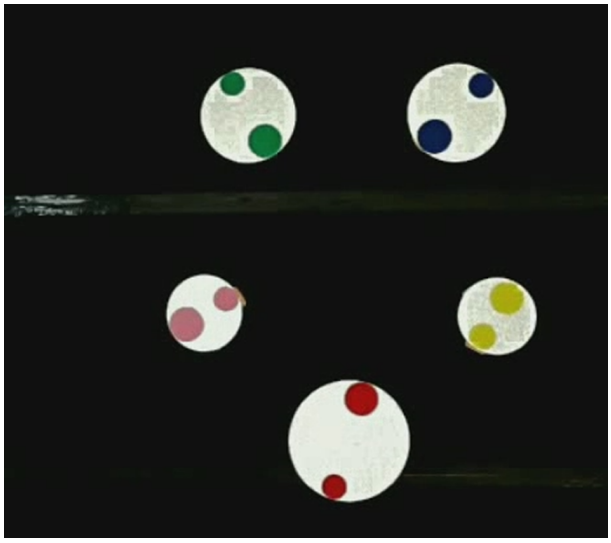


Table 9: *Performance Metrics for Shape - 3*

Total Displacements[m]	Settling Time[sec]
1.95	26

Figure 57: *Formation Shape 4- Matlab Environment*

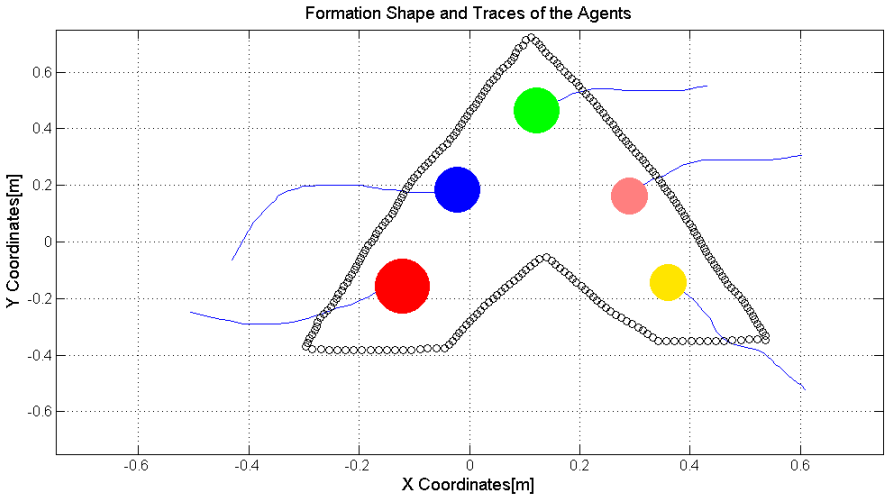


Figure 58: *Formation Shape 4- Test Environment*

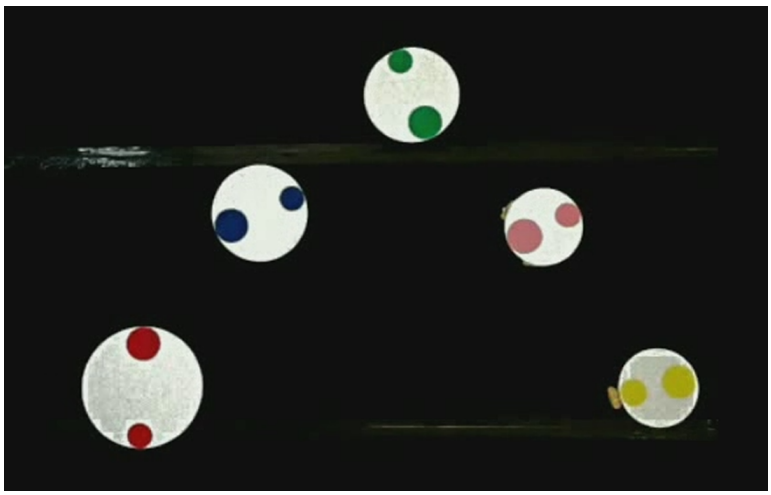


Table 10: *Performance Metrics for Shape - 4*

Total Displacements[m]	Settling Time[sec]
2.16	29

Figure 59: *Formation Shape 5- Matlab Environment*

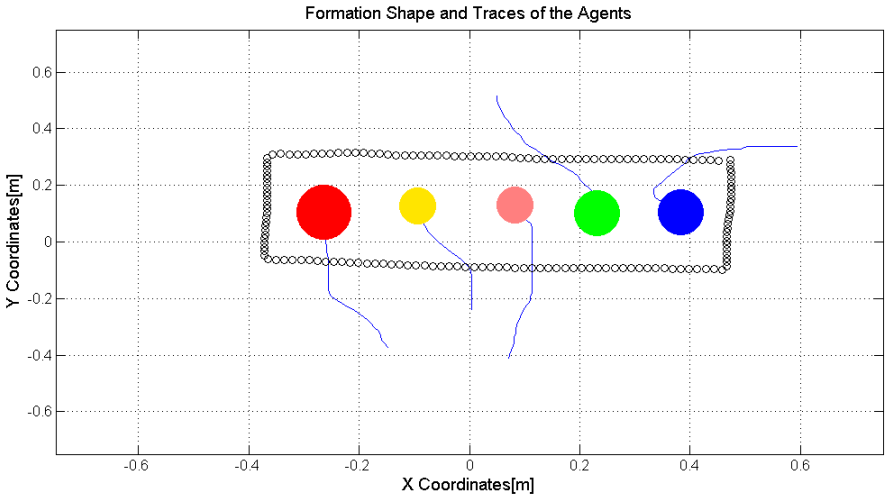


Figure 60: *Formation Shape 5- Test Environment*

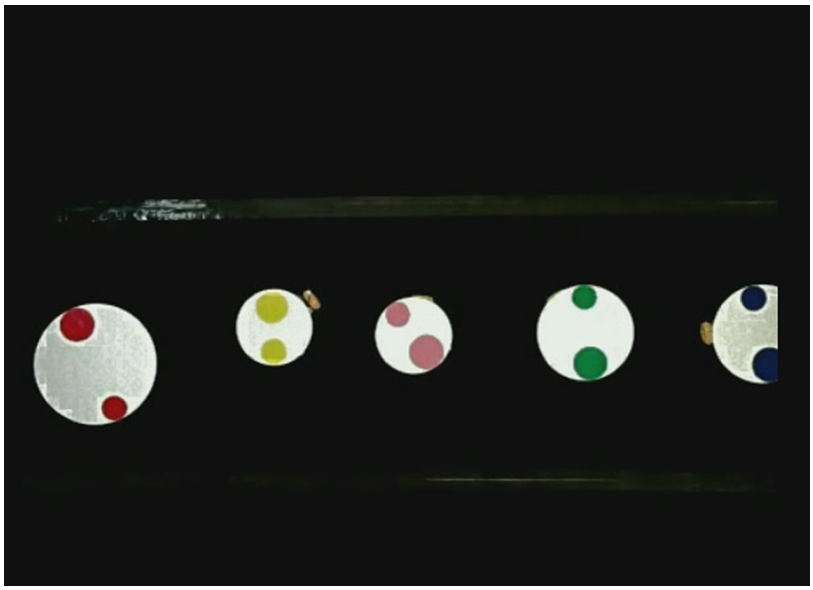


Table 11: *Performance Metrics for Shape - 5*

Total Displacements[m]	Settling Time[sec]
1.80	25

VI. Conclusion and Future Works

I. Conclusion

In this thesis work, we have proposed solutions to the dynamic formation control of heterogenous mobile robots. The solution is constructed with the help of two different subsystems, local positioning systems and formation control systems. A local positioning system is designed to provide agents accurate position and velocity data. It is assumed that only a percentage of the swarm has position measurement sensors (e.g. a GNSS solution) due to the low sensor capabilites which is discussed in Section-xx. So there is a nececssity to implement a solution for estimating the positions and the velocities of the rest of the agents which do not have external position measurement sensors on their board. A solution proposed including the fusion of inertial measurements of the agents and the position data calculated with the help of the position beacons (i.e. the agents which have position measurement sensors on their boards). An algorithm that provides INS with the help of IMU sensors (i.e. Accelerometers, Gyroscopes, Magnetometers) is designed for each agent which propogates the translational acceleration of the agents in the Earth frame by implementing an AHRS system. Since this INS solution is vulnerable in position and velocity estimations due to the drift problems and measurement noise of the sensors, an external measurement of position is provided to each agent to correct their estimations based on local trilateration process. Local trilaterion process includes the distribution process of the position data from the position beacons to the agents which do not have position sensors. Since it is assumed that the agents have limited communication capabilities, a solution to distribute the position data with minimum cumulative error to the agents farthest from the position beacons is provided with the help of route tables and assignment of rank values to the agents in the mesh network. Trilaterion process is handled with the local neighbors of each agent to calculate the estimated position data. This position data is used in the Kalman observers as the external measurement data. Since it is not possible to execute the route table determination and the local trilateration process with the main process period of the agents (which controls the actuators, propogates INS measurements etc.) due to the computational complexity of the algorithm, a maximum period of time to execute localization process is determined with the help of an upper error bound allowable to achieve formation control with success. The maximum error norm for each agent is calculated below 3 meters with Monte Carlo simulations with the localization period of 3 seconds and this error value is chosen as the maximum tolerable value for the formation control system. So the period for the localization timer is chosen as 3 seconds.

In the second subsystem of the solution, we have introduced two types of methods based on potential field based approach and shape partitioning based approaches. Potential field based approach implements virtual forces to the agents created by the desired formation shape, obstacle in the environment and the other agents. The X-Swarm theory is introduced to prove that the movement of an agent which is outside of the desired formation shape will be towards to the center of the center of the swarm. The potential field based algorithm is designed to satisfy the conditions for the X-Swarm to direct the agents to the center of the swarm which is also moving towards to the center of the formation shape. It is observed that the agents are capable of homogenously covering of the desired formation shape with the help of these artificial forces. The other two methods based on a shape partitioning approaches, bubble packing method and randomized fractals method . These methods have different solutions to partition the desired formation shape into potential goal states. The algorithms which determines the assignment of the agents to these goal states in a decentralized manner is identical for these two methods. Bubble packing algorithm introduces a method to partition the desired formation shape based on the Van der Waals forces between the molecular bonds to distribute the bubbles homogenously. It is widely

used in mesh generation problems. The idea is to generate a mesh for a surface with identicak bubbles to minc a regulare Voronoi diagram including the vertices representing the center of the bubbles. In our problem, the agents are represented by the bubbles with radius' representing their coverage circles described in Section-xx and an offline simulation handled to generate the desired homogenous mesh to partition the shape into potential goal states. Each goal state has the information of agent type (i.e. coverage circle radius) to be placed at the final configuration. The bubble packing method have a similar approach with the potential fields since both implements intermember&interbubble forces to homogenously distribute the agents&bubbles in the desired formation&surface

Randomized fractals method based on distributing the coverage circles which represent the different agent types in the desired formation shape, This algorithm is faster than the bubble packing method since it is not executing an offline simulation.

The assignment processof the agents to the potential goal states is identical for these two different shape partitioning methods. First, the obstacles in the environment is augmented with Minkowski sums for different types of agents. The total coverage of the augmented obstacles represents the forbidden space for each agent. The free configuration space is calculated by extracting these forbidden spaces from the configuration space itself. Then each agent calculates the visibility graphs in the environment with augmented obstacle by including its current position and the potential goal states determined by the shape partitioning algorithms. To find the shortest paths in these visibility graphs to the each potential goal state is calculated with the help of Dijkstra's Algorithm. Hungarian algorithms is implemented to reach a global consensus on the assignment of the goal states to the agenst based on minimizing the total energy consumption (i.e. total displacements) by taken into consideration the distances of shortest paths to each goal states reported by each agent.

Monte carlo simulations with 1000 iterations is held with the same initial conditions and formation shapes for three proposed solution. To evaluate the performance if these methods, mesh quality, total energy consumption and settling time metrics are calculated. As expected the worst mesh quality performance belongs to the randomized fractals method since it has an approach to randomly distributes the coverage circles in the desired formation shape. The potential field based method and the bubble packing method have similar mesh qualities because of their nature implementing intermember&interbubble forces which contibutes the homogenously distrubution of the agents&bubbles in the desired formation shape. On the other hand, potential field based method have the worst settling time and total energy consumption performance due to the lack of predetermined goal states for the agents in the desired formation shapes. The best performance is achieved with the bubble packing method in terms of three metrics and hence some different formation shape trials including special cases like a formation shape which includes obstacles are held with this method.

At the end of the work, a hardware implementation of bubble packing method is held with 5 mobile robots with different sizes. Since this implementation is done at the indoor environment and the number of agents is not sufficient, it is not possible to implement the local positioning system at this demonstration. The position and orientation data for the agents are calculated with image processing algorithms based on visual feedback provided by an E/O camera. A mesh network established with Zigbee modules are used to provide a communication backbone between the agents and the mission computer. The agents are designed as 3 axis omni-wheel mobile robots which has the capability to translate in the environment without the need for adapting its orientation. Each agent has its own processor module to execute the goal state decision process, control algorithms and step motor control routines. This feature demonstrates the decentralized manner of the solution in which there is no central server deciding the final states of each agent

in the solution. Several formation shape trials are done and the performance is evaluated with total energy consumption and settling time metrics. It is observed that the agents are capable to achieve different formation shapes as desired. This work is important to show the proposed solution about the formation control problem can be implemented in real time in the environment. The results represent a proof of concept (POC) of the thesis work rather than implementing the all details of the proposed solution.

II. Future Works

It is important to realize the proposed solution with more agents within the environment. This kind of implementation will give a more insight about the performance and the drawbacks about the system according to the tests held in simulation environment. So, the implementation with more number of agents in real time is one of the next steps for this project.

Even if the bubble packing method has the best performance in the sense of both mesh quality, energy consumption and settling time issues, it is not a fully decentralized method since the shape partitioning process is executed on a central server node. Shape partitioning algorithm is not deterministic and it doesn't converge to the same potential goal states even if the initial states are identical. If it is possible to implement a method to partition the desired formation shape with a similar mesh quality performance in a more deterministic manner, it will be possible to distribute the partitioning process to individual agents.

Since the main focus is not on the obstacle avoidance issue of the agents, all three methods are augmented with only potential fields by the obstacles in the environment just to implement a basic obstacle avoidance. It is needed to implement a more complex obstacle avoidance algorithm (e.g. tangent bug algorithm) to avoid the cases in which some of the agents gets into an unwanted equilibrium under the effect of the desired setpoints and the obstacle forces.

In this thesis work, one of the main metrics to evaluate the performance of the proposed solutions is the total energy consumption of the swarm while achieving the desired formation shape. Moreover, in the shape partitioning methods the goal state assignment algorithms are designed to minimize the overall energy consumption of the swarm. But, there may be cases in which some of the agents are more critical to reach the goal state or some of the agents are running out of battery&energy. In such conditions, it will be appropriate to improve the goal state assignment algorithm to prioritize the agents to handle these kind of special cases.

The main aim was to create a swarm with heterogenous agents which can achieve a desired complex formation. One of the idea about this concept is to achieve some complex tasks collectively with the agents which have different individual capabilities. In our thesis work, we have focused on achieving complex formation shapes with heterogenous mobile robots rather than achieving complex tasks after getting the desired formation shapes. The next step for this project is to add the capability of doing these kind of tasks with the agents from different capabilities.

VII. ornekler

$$k_t \text{ and } k_r \tag{111}$$

$$A_i = C_{free}(R_i, S) + C_{forb}(R_i, S) \tag{112}$$

$$S_1 \oplus S_2 := \{p + q : p \subset S_1, q \subset S_2\} \tag{113}$$

$$F_{i,m,x} = k_m \sum_{j=1, j\neq i}^{n} \left(\frac{x_i - x_j}{d_{ij}} \frac{1}{\left(d_{ij} - d_o\right)^2} \right) \tag{114}$$

$$X_i = \begin{bmatrix} z_i \\ \dot{z}_i \end{bmatrix} \tag{115}$$