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

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

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;

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

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

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

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. ORNEKLER

$$A_i = C_{free}(R_i, S) + C_{forb}(R_i, S) \tag{45}$$

$$S_1 \oplus S_2 := \{p + q : p \subset S_1, q \subset S_2\} \tag{46}$$

$$F_{i,m,x} = k_m \sum_{j=1, j \neq i}^{n} \left(\frac{x_i - x_j}{d_{ij}} \frac{1}{(d_{ij} - d_o)^2} \right) \tag{47}$$

$$X_i = \begin{bmatrix} z_i \\ \dot{z}_i \end{bmatrix} \tag{48}$$