Consensus based Rendezvous Problem for Multi-Robot Systems

Poojan Patel, Varadaraya Ganesh Shenoy

December 4, 2020

1 Abstract

Consensus control and coordination is a vital repetitive task. This study aims to provide an example of the consensus algorithm used to implement rendezvous application for multi-robot scenarios. In scenarios that involve multiple mobile robots, rendezvous is seen as a great opportunity to share information between robots. Although pre-defined points of rendezvous are popular, they do not provide flexibility in case of communication/robot failure and sudden changes in the environment as the rendezvous point will then be compromised. Consensus-based rendezvous allows for deciding the meeting point on-line, therefore removing the effects of the environment and establish constant communication between robots. But most assumed that sensor measurements are free from noise which is not true on most occasions.

[3]The consensus-based rendezvous problem is very helpful in applications such as exploration where the network topology is dynamic. For mapping applications, this study helps in allowing the robots to find an agreed place to share each of the robot's maps, therefore covering larger areas. Consensus-based controllers are robust to noise as all sensors are susceptible to measurement noise. Wireless devices and received signal strength (RSS) measurements have recently become popular in mobile robot scenarios. Notably, the direction of arrival (DOA) of the RSS can be estimated using directional and omnidirectional antennas. [10]

[10] proposes wireless signal measurements to estimate the direction of arrival (relative bearings) of neighboring robots. We realize this objective by designing a consensus controller using receiver signal strength measurements and the direction of arrival of wireless signals.

This project is carried out using Robotarium in MATLAB. Graph-based modeling and consensus algorithms help us illustrate an accurate model. Energy and convergence of each model are the properties that are examined. Also, the iteration at which the robots reach a minimum distance from each other is seen. From the results found, it is fairly evident RSS-DOA addition to the bearing algorithm achieves faster convergence and the energy peak is much lesser than bearing only and co-ordinates based consensus algorithms.

2 Division of Labor

Poojan Patel: - Modeling and Simulation of the consensus-based Rendezvous application.
Varadaraya Ganesh Shenoy: - Equilibrium analysis and Controller Design for the consensus-based Rendezvous application.

3 Mathematical Modelling

Mathematical Model: - The rendezvous control problem of multi-robot systems inspired by some papers [1,2,3,4]. As the development of the wireless communication has advance the robotics to next level, we have assumed to use the wireless communication between the robots for the information transfer. To achieve the consensus position (e.g. using geometric features such as circumcenter) most of the studies have use the sharing techniques for the robot states (e.g. relative position in global coordinate frame) .

Relatively few studies have addressed the rendezvous problem without sharing coordinates [[12], [14]]. In such cases, local sensors such as vision can be used to detect and identify neighboring robots and to direct the rendezvous process in a distributed fashion [1].

We have used a novel distributed bearing aided rendezvous control using DOA and assuming RSS measurements from wireless devices fabricated on the mobile robots as an alternative sensing modality for bearing estimation. In [15], the proposed control law assumed pseudo-positions for the neighbor robots based on their relative bearings and the sensing range. Then, the robots moved towards the center of the smallest circle that encompassed all neighbor robot pseudo-positions. In addition, coordinate-free formation (shape) controllers that do not need an external positioning or coordinate referencing system have been well-studied [[9], [14]].

Modeling Approach: - Graph Theory used to represent the 12 mobile robots' connectivity. The Graph $\mathbb{G}$ is connected.

[5]Graph Theory: - The fixed, directed graph $\mathbb{G}$ below represented the communication network of a group of 15 robots, denoted by circles. The weights of the label is represented in the edges with the value one.

Problem Statement:Consider a set of N non-holonomic mobile robots in workspace W $\subset$ R_d (we assume d = 2 in this paper) with their state denoted as (position) $q_i = [xi, yi]^T \epsilon R_2$ and (orientation) $\theta_i, i\epsilon V = [1, 2, ..., N]$, defined in the inertial (world) frame F_W. Their unicycle kinematics is governed by:

$$\begin{aligned} \dot{x_i} &= u_i cos\theta_i \\ \dot{y_i} &= u_i sin\theta_i \\ \dot{\theta}_i &= \omega_i \end{aligned} \tag{1}$$

where $u_i \epsilon$ R is the linear velocity input, and $\theta_i \epsilon$ R is the angular velocity input. Let us suppose that the robot i has a local coordinate frame F_i that is attached to F_W and the rotation matrix that maps the local coordinate frame to the global frame is given by :

$$R(\theta_i) = \begin{bmatrix} cos\theta_i & sin\theta_i \\ -sin\theta_i & cos\theta_i \end{bmatrix}$$

Let $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ denotes an undirected weighted graph with set of vertices (robot nodes) $\mathbb{V}\epsilon R_N$, edges set $E = (i, j) \subset \mathbb{V}\mathbb{V}i\epsilon N_j$. The adjacency matrix is $A = [a_{ij}]NN$. For each robot i, its set of neighbors is denoted as $N_i = j\epsilon\mathbb{V}|(i, j)\epsilon E$, and the cardinality of the set N_i (number of neighbors) is $|N_i|\epsilon R_N$. The weights $a_{ij} = 1$, if (i,j) ϵ E (if there is a connection between i and j). otherwise, $a_{ij} = 0$. We assume the following properties of the graph: G is symmetric (i.e. $a_{ij} = a_{ji}$); there is no self-loop and repetitions of edges (i.e. $a_{ii} = 0$); and G is bidirectional (i.e. $i\epsilon N_j \leftrightarrow j\epsilon N_i$).

- Assumption 1: A robot senses and communicates with another robot when their relative distance is within the sensing range, i.e. $k_{qj}q_{ik} \leq S_{max} \Rightarrow j\epsilon N_i$.
- Definition 1: The graph G is connected if there is a communication path from every robot to every other robot in V as per Definition 1.
- Assumption 2: (Merge) Two robots merge into one if the distance between them is within a small threshold ϵ R+.
- Definition 2: (Consensus). The robots achieve rendezvous if all their positions merge to a common point over time, i.e. when

$$\lim_{t\to\infty}(q_iq_j) = 0, \forall i\ j\epsilon V.$$

- Definition 3: (Coordinate-free). A control law is coordinate-free if each robot can work independently with its local coordinate frame of reference [2].
- Assumption 3 The connectivity (interaction) graph G is initially connected as per Definition 1, but not necessarily fully (completely) connected, i.e. $N_i \geq 1\forall i\ \epsilon$ V.

4 Theoretical Analysis

The above model is analyzed for 3 properties namely: equilibrium robot configuration (convergence to a common location), stability characteristics of the equilibrium and connectivity maintenance. [10] defines two theorems which help us analyze the three properties.

Theorem 1(Convergence and Stability): *Under the control law in Eq. (7) with the noisy RSS measurements, the group of robots converges to a single point within a small threshold over time (Definition 2) and achieves rendezvous if Assumption 3 holds true*

Proof: Stability of the nonlinear system stability without noise is proven first.Later, we incorporate noise in our analysis through a discretized dynamical system.

Consider that the measured bearing is noise-free,ϕ_j. To prove the convergence, we follow Lyapunov control approach similar to [[15], [14]]. We need to prove that for each robot i, the sum of relative distances between all its neighbors $\sum_{j\epsilon N_i}\|q_j - q_i\| \implies 0\ \forall$i ϵ V as t $\to \infty$. Using this as a Lyapunov candidate function, the total displacement Energy value is given by,

$$V = \sum_{i\epsilon\mathbb{V}}\sum_{j\epsilon N_i}\|.q_{ij}\| = \sum_{i\epsilon\mathbb{V}}\sum_{j\epsilon\mathbb{V}}a_{ij}\|.q_j - q_i\| \tag{2}$$

Here, the adjacency index $a_{ij} = 1$ if robots i and j are neighbors, and $a_{ij} = 0$ otherwise. Using Eq. (1), we have the time derivative of V as:

$$\dot{V} = -2 \sum_{i} \sum_{j \epsilon \mathbb{V}} a_{ij} \frac{(q_j - q_i)^T}{\|.q_j - q_i\|} \dot{q}_i \tag{3}$$

After some manipulations with the help of Eq. (8) and (1), we obtain:

$$\dot{V} = -2 \sum_{i \epsilon \mathbb{V}} \sum_{j \epsilon \mathbb{V}} a_{ij} u_i cos\phi_{ij}^w \tag{4}$$

$$\dot{V} = -2 \sum_{i \epsilon \mathbb{V}} u_i \sum_{j \epsilon \mathbb{V}} a_{ij} cos\phi_{ij}^w \tag{5}$$

Thus, we show that the Lyapunov function is negative semidefinite. The derivative $\dot{V}$ is 0 only when the velocity $u_i = 0$ for all robots i ϵ $\mathbb{V}$. The largest invariance set (with assumption 2) is V = 0 when $u_i = 0$, which can occur even when $\|q_{ij}\| \neq 0$ (not all robots converged at equilibrium point with V = 0). However, when $u_i = 0$ and $\|q_{ij}\| \neq 0$, then $\omega_i \neq 0$ because the projection matrix of $_i$ is orthogonal to that of u_i of Eq.(7) and (6).This will eventually drive at least one i robot in the system to a different state where $u_i \neq 0$, thus making V < 0. Also, the weights asymptotically decay with the distance. Therefore, according to LaSalle invariance principle [7], the system asymptotically converges to a common state and achieve rendezvous. This concludes the first part of the proof.

Now assume a noisy measurements as in Eq. (2) for DOA. Following the approach in [4], we use the condition that the expected value of stochastic control input caused by the measurement noise has the same sign as that of the noise-free (deterministic) control input .i.e. $\mathbb{E}[\tilde{u_i}].u_i \geq 0$.Through the Lyapunov analysis of the discrete dynamical system, we have

$$\dot{V} = -2 \sum_{i \epsilon \mathbb{V}} |N_i| (\sum_{j \epsilon \mathbb{V}} a_{ij} \mathbb{E}[\omega_{ij}] cos\phi_{ij}^w)(\sum_{j \epsilon \mathbb{V}} a_{ij} \mathbb{E}[\omega_{ij}] cos\phi_{ij}^w) \leq 0 \tag{6}$$

Although both ω_{ij} and bearing ϕ_{ij} depend on the RSS measurements, they are independent of each other because of the different time instants when the measurements are made. For instance, for the bearing estimation, the RSS measured in all the sensors or rotation angles are used, whereas for weights only one instant of the measurements are used.This potentially results in $\mathbb{E}[\tilde{u_i}].u_i \geq 0$. Therefore,the noise will not affect the system stability. This concludes the proof.

Theorem 2(Connectivity Maintenance) : *Under the control law and assumptions. The graph G remains connected throughout G remains conntected throughout rendevous process.*

Proof: To prove this, we use the same approach as the Preposition 5.1 of [13]. The weights w_{ij} in Eq. (6) act as the artificial potential field to the controller. The R_{th} is defined by the sensing range S_{max} (in our case, the sensitivity of the wireless receiver). Both R_{th} and R_{ji} in w_{ij} depend on the distance between the robots and therefore, are indirect

representation of distance measurements. This enables us equate our controller weights to the distance-based weights analyzed in [19]. We have that,

$$\lim_{\|q_{ij}\| \to S_{max}} \omega_{ij} = \infty \tag{7}$$

Although we do not explicitly state the proof here, we can show that the derivative of the potential field after introducing a hysteresis into the system is non positive (i.e.$\dot{\omega_{ij}} \leq 0$), similar to [13]. Thus, all the links in G are maintained, which ensures that the graph is always connected at all instants during the rendezvous process.

4.1 Designing a weighted bearing rendezvous controller

Wireless network topology is used to build the interaction graph $\mathbb{G}$.
Controller:

$$\begin{bmatrix} u_i \\ \omega_i \end{bmatrix} = \frac{v_{max}}{|N_i|} \sum_{j \epsilon N_i} \omega_{ij} \begin{bmatrix} cos\phi_j^i \\ sin\phi_j^i \end{bmatrix} \tag{8}$$

$$\omega_{ij} = \frac{1}{-(R_{th} - R_j^i)}$$

where R_{th} is the threshold RSS power, R_{ji} is the RSS measured at robot i coming from the AP of robot j, and ϕ_{ij} is the DOA measured by robot i from j in the local frame F_i (Sec. 4.1) because the controller is expressed in the local frame F_i. The controller can be applied in the global frame F_W (single-integrator dynamics) using the rotational transformation as

$$\begin{bmatrix} cos\tilde{\phi}_{ij}^W \\ sin\tilde{\phi}_{ij}^W \end{bmatrix} = R(\theta_i) \frac{v_{max}}{|N_i|} \sum_{j \epsilon N_i} \omega_{ij} \begin{bmatrix} cos\phi_j^i \\ sin\phi_j^i \end{bmatrix} \tag{9}$$

Note, since the RSS is a naturally decreasing function of the inter-robot distance $\|q_{ij}(t)\|$, the weights w_{ij} are state-dependent. However, the weights are normalized to bound the command inputs and to minimize the impact of geometric bias and sensor noise among the RSS measurements of the neighbor robots.

5 Numerical Validation

The proposed controller for rendezvous is validated by conducting simulations using the Robotarium framework in MATLAB [11]. Robotarium is chosen because it provides a barrier certificate utility function for collision avoidance of the robots. But, this may affect the efficiency of the execution of the algorithms. Therefore, it is not used in the simulations so as to achieve a fair comparison.

The RSS measurements that are used in our code , have parameters that are usually encountered in wireless network: $RSS_{d_0} = -20dBm$, $\eta = 3$ and $\lambda = 4dBm$ [8]. A scaling factor 50 is applied such that the simulation area becomes (150m x 150m), therefore obtaining realistic simulations. The sensing range is set to $S_{max} = 0.8m$ [10].

5.1 Scenario

The proposed controller is verified with co-ordinates based method, and bearing only method with random initial positions of robots. For each controller, Robotarium is iterated 1000 times.

5.2 Comparison with SOTA Methods

The proposed rendevous controller is compared with the following SOTA methods.

- Co-ordinates based [1] : The most popularly used consensus controller methods that uses robot co-ordinates. It is hypothesized that the proposed controller has performance similar to this method.
- Bearing-only controller [6] : This bearing based controller uses enclosing circles.

5.3 Evaluation Metrics

To evaluate the performace of each consensus controller , the following metrics are used :

- Energy: Lyapunov Candidate Function in Eq.(9).
- Speed of Convergence: The number of iterations taken to achieve the stop condition (minimum distance between each robot is 0.5m)
- Connectivity preservation: Whether the final graph is connected or not.

6 Results

The following table depicts the results obtained :

Algorithm	Converging iteration	Stop-distance iteration
RSS-DOA	500	337
Co-ordinates preserving	600	299
Bearing-only enclosing circles	800	249

Table 1: Results

The Laplacian Matrix for the model is as shown :

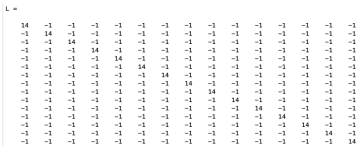

Figure 1: Lapalacian Matrix

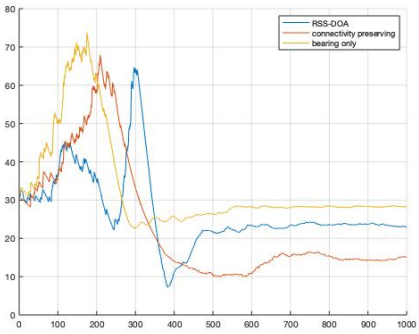

Figure 2: Iteration vs Energy

From the results above, it is clear that although the RSS-DOA takes more iterations to reach the stop-condition, it reaches minimum much quicker than the other two SOTA methods.

References

[1] Lin J. Morse A.S.A adbabaie, A. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *Proceedings of the 41st IEEE Conference on Decision and Control, 2002*, 3:2953–2958, 2002.

[2] López-Nicoĺs G. Sagus Aranda, M. Coordinate-free control of multirobot formations. *Springer*, page 131–181, 2017.

[3] C. H. Caicedo-Nunez and M. Zefran. Consensus-based rendezvous. *2008 IEEE International Conference on Control Applications, San Antonio, TX*, pages 1031–1036, 2008.

[4] Martinoli-A. Gowal, S. Bayesian rendezvous for distributed robotic systems. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[5] Victor M. Zavala Jordan Jalving, Yankai Cao. Graph-based modeling and simulation of complex systems. 125:134–154, 2019.

[6] Digumarti S.T. Oung R. D'Andrea R. Kriegleder, M. Rendezvous with bearing-only information and limited sensing range. *2015 IEEE International Conference on Robotics and Automation (ICRA),*, page 5941–5947, 2015.

[7] J.P La Salle. The stability of dynamical systems. *SIAM*, 1976.

[8] Johansson K.H. Bicchi A Lindhe , M. An experimental study of exploiting multipath fading for robot communications. *3rd International Conference on Robotics Science and Systems, RSS 2007, 27-30 June 2007, Atlanta, GA, USA*, page 289–296, 2008.

[9] Cristofalo E. Zhou D. Schwager M. Sagus Montijano, E. Vision-based distributed formation control without an external positioning system. *IEEE Transactions on Robotics*, 32(2):339–351, 2016.

[10] Ramviyas Parasuraman and Byung-Cheol Min. Consensus control of distributed robots using direction of arrival of wireless signals.

[11] Glotfelter P. Wang L. Mote M. Ames A. Feron E. Egerstedt M. Pickem, D. The robotarium: A remotely accessible swarm robotics research testbed. *Robotics and Automation (ICRA), 2017 IEEE International Conference*, page 1699–1706, 2017.

[12] LaValle S.M. Liberzon D. Yu, J. Rendezvous without coordinates. *IEEE Transactions on Automatic Control*, 57(2):421–434, 2012.

[13] Egerstedt M. Pappas G Zavlanos, M. Graph theoretic connectivity control of mobile robot networks. *Proceedings of IEEE*, 99, 2011.

[14] Zheng R. Zhao, S. Flexible bearing-only rendezvous control of mobile robots. *Chinese Control Conference (CCC)*, page 8051–8056, 2017.

[15] Sun D. Zheng, R. Rendezvous of unicycles: A bearings-only and perimeter shortening approach. *Systems Control Letters*, 62(5):401–407, 2013.

Required for the code

```
%% This is the main file to conduct experiments with
several consensus control (rendezvous) algorithms of N
robots in the Robotarium testbed
%% Number of robots
function energy =
rendezvous_experiments(N,algorithm,iterations)

global sensing_range error_bearing error_distance
uni_to_si_states si_to_uni_dyn G N desired_distance;

fH = str2func(algorithm); % function handle for the chosen
rendezvous algorithm

%% Get Robotarium object used to communicate with the
robots/simulator
rb = Robotarium('NumberofRobots',N);
% Build the Robotarium simulator object!
figure_robotarium = figure(1); movegui('northeast');
movegui('onscreen');
title('Rendezvous algorithm experimented in Robotarium
testbed');

%% Experiment parameters
desired_distance = 0.5; % desired inter-agent distance
range to realize stop condition
sensing_range = 0.8; % Sensing radius within which robot i
detects robot j (same for all the robots)
error_bearing = 0.1; % Standard deviations of the bearing
measurment error (radians)
error_distance = 0.1; % Standard deviations of the distance
measurment error (m)
root_node = 5; % Set the root node robot where the
rendezvous should occur (instead of a consensus point), if
rendezvous_at_root flag is set/used below
desired_energy =  10; % (currently not used) desired value
of the Lyapunov candidate energy function
safety_radius = 0.11; % Do NOT change this - set for
Robotarium. It is the safety radius for collision avoidance
between robots
dxmax = 0.075; % Do NOT change this - set for Robotarium.
The maximum velocity of the GRITSbots robots (<=0.8m/s)
(used if normalize_velocities is set/used below)

%% Flags to use specific parts of the code
```

```
collision_avoidance = 0; % To enable/disable barrier
certificates
normalize_velocities = 1; % To normalize the velocities
(recommended - Do NOT reset)
initial_positions_fixed = 1; % If set, then fixed initial
positions (defined further down) will be used instead of
random
update_network_topology = 1; % To enable/disable the update
of connected graph (dynamically) in every iteration
rendezvous_at_root = 0; % If set, it will make all the
robots rendezvou at a root node (defined below) instead of
a common location
formation_control = 0; % If set, the robots will create a
specified formation (shape) instead of rendezvous
% Flags for Plotting tools
plot_initial_graph = 1; % To plot initial connected graph
plot_dynamic_graph = 1; % To plot updated connected graph
in every iteration
plot_robot_index = 1; % To enable/disable the display of
robot index on top of each robot in the Robotarium figure
plot_robot_trajectory = 0; % To enable/disable the display
of robot trajectory in the Robotarium figure
plot_robot_initialposition = 1; % To enable/disable the
display of robot initial position in the Robotarium figure
make_movie = 0; % To create a movie using the Robotarium
figure over the iterations

%% Formation control parameters (if formation_control flag
is set)
% to replicate a P-like formation
b = floor(N/2);
c = N-floor(N/2);
angle = 0:(360/c):359;
target_x = 0.5*cos(pi)*ones(1,b) ;
target_y = 0:-1.4/(b-1):-1.4;
target_configuration =
[[target_x,0.5*cos(angle*pi/180)];[target_y,0.5*sin(angle*p
i/180)]];
% to replicate a circular formation
%angle = 0:(360/N):359;
%target_configuration =
[0.5*cos(angle*pi/180);0.5*sin(angle*pi/180)];
```

```
%% Grab tools we need to convert from single-integrator to
unicycle dynamics
%Gains for the transformation from single-integrator to
unicycle dynamics
linearVelocityGain = 2; %1
angularVelocityGain = pi;
transformation_gain = 0.06;

% Gain for the diffeomorphism transformation between
single-integrator and
% unicycle dynamics
[~, uni_to_si_states] =
create_si_to_uni_mapping('ProjectionDistance',
transformation_gain);
si_to_uni_dyn =
create_si_to_uni_dynamics_with_backwards_motion('LinearVelo
cityGain', linearVelocityGain, 'AngularVelocityLimit',
angularVelocityGain);
% Single-integrator position controller
si_pos_controller =
create_si_position_controller('XVelocityGain', 2,
'YVelocityGain', 2);
% Collision avoidance - barrier certificates
si_barrier_cert =
create_si_barrier_certificate('SafetyRadius',
safety_radius);

%% Initialize the robots to fixed positions (if
initial_positions_fixed is set)
if(initial_positions_fixed == 1)
    %initial_positions = [-0.2 -0.5 0.5 0.4 -0.1; 0.2 -0.4
-0.1 0.6 0.7]; % for N=5
    initial_positions = [0 0.4 0.5 0.4 -0.1 -0.3 -0.5 -0.7
0 1 -1 -1 0.3 -0.5 0.9; 0.3 0.9 1.1 -1 -0.2 -0.9 -0.3 -1
1.2 -1.2 0.2 -0.9 -0.4 0.6 1]; % N=15
    %initial_positions = [0 0.4 0.5 0.6 -0.1 -0.3 -0.5 -0.7
0 0.8 -1.1 -1 0.3 -0.5 0.9 0.5 -0.5 -1 1 1.4; 0.3 0.9 1.1 -
1 -0.2 -0.9 -0.3 -1 1.2 -1.2 0.2 -0.9 -0.4 0.6 1 0.5 1.3 1
-0.2 0.2]; % for N=20
    r = initialize_robot_positions(rb,N,initial_positions);

end
```

```
%% Finding the connected graph (Also checking if the
initial interation/network graph is connected)
x = r.get_poses();
xi = uni_to_si_states(x);
r.step();
[L,G] = GetConnectedGraph(x(1:2,:),sensing_range); %
Getting the initial connected Graph
if(length(find(eig(L) <=0.001)) > 1)
    connection_status_initial = 0; % disconnected graph
    disp('Initial graph has some disconnected components.
For experiments, we need a connected graph. Please try
again!');
    return; % If it's a disconnected graph, then stop the
script
end

%% Initiating connected graph figure window
if(plot_initial_graph == 1)
    figure_graph = figure(2); plot(G); title('Initial
Network/Interaction Topology');
    xlim([-3 3]);
    ylim([-3 3]);
    movegui('northwest');
end

%% Experiment initialization
max_iterations = iterations; % the number of iterations for
the experiment

% Initialize velocity vector for agents.  Each agent
expects a 2 x 1
% velocity vector containing the linear and angular
velocity, respectively.
dxi = zeros(2, N);

previous_xi = xi; % A temporary variable to store the
position values
distance_travelled = zeros(1,N); % total distance traveled
by each robot - Performance evaluation metric
iteration_at_stopcondition = 0; % number of iteration at
which the stop condition is reached
iteration_at_minenergy = 0; % number of iteration at which
the energy function values is the minimum (less than a
threshold)
```

```
energy = zeros(1,max_iterations); % The value of the Energy
function which is sum of all distances between the
connected nodes
mycols = jet(N); % To display colored trajectory for each
robot (if plot_robot_trajectory is set)
fig_index = figure_robotarium;
fig_traj = figure_robotarium;
Movieframe(1:max_iterations) = gcf; % to be used in movie
creation if make_movie flag is set/used

%% Display the robot's initial position trajectory in the
Robotarium figure
if(plot_robot_initialposition == 1)
    %fig_traj = set(0,'CurrentFigure',r.figure_handle);
    for i=1:N
        fig_traj =
plot(x(1,i),x(2,i),'o','Color',mycols(i,:));
    end
end

disp('Rendezvous process initiated');

%% Display the title text in the Robotarium figure
set(0,'CurrentFigure',r.figure_handle);
alg_string = regexprep(algorithm,'_',' ');
alg_string = regexprep(alg_string,'\s*.','${upper($0)}');
%alg_string = strcat('Algorithm: ',alg_string);
text(-
1.4,1.4,alg_string,'FontSize',8,'Color','red','FontWeight',
'Bold', 'Interpreter', 'none');

%% Iteration starts here (for the previously specified
number of iterations)
for t = 1:max_iterations
    %disp(t) % to display the iteration number
    stop_condition = 1; % This variable is to define the
stop condition. If it's 1 - then stop the
iteration/experiment

    set(0,'CurrentFigure',r.figure_handle);
    fig_iter = text(-1.4,-1.4,strcat('Iteration :','
',num2str(t)),'FontSize',10,'Color','red','FontWeight','Bol
d');
```

```
    % Retrieve the most recent poses from the Robotarium.
The time delay is
    % approximately 0.033 seconds in Robotarium
    x = r.get_poses(); % Get unicycle coordinates
(x,y,theta)
    xi = uni_to_si_states(x); % convert the unicycle pose
to SI units (x,y)

    %% Formation control (if formation_control is set/used)
    if(formation_control == 1)
        xi = xi - target_configuration;
    end

    %% Display the robot's index on top of each robot in
the Robotarium
    if(plot_robot_index == 1)
        for i=1:N
            set(0,'CurrentFigure',r.figure_handle);
            fig_index(i) =
text(x(1,i),x(2,i)+0.04,num2str(i),'FontSize',6,'Color','re
d','FontWeight','Bold');
        end
    end

    %% Display the robot's trajectory in the Robotarium
figure
    if(plot_robot_trajectory == 1)
        for i=1:N
            set(0,'CurrentFigure',r.figure_handle);
            fig_traj(i) = plot(x(1,i),x(2,i),'.--
','Color',mycols(i,:));
        end
    end

    %% Update the connected tree dynamically
    if (update_network_topology == 1)
        [L,G] = GetConnectedGraph(x(1:2,:),sensing_range);
% Finding the initial connected Graph
    end

    %% Consensus/Rendezvous Algorithm
    [dxi,stop_condition,energy(t)] = fH(L,xi); % The
rendezvous algorithm chosen in the beginning
```

```
    %% Change the Rendezvous location (if need to rendezvou
at a specific root node)
    if(rendezvous_at_root == 1 && formation_control ~= 1) %
It can work only for rendezvous and not for formation
control
        dxi(1,:) = dxi(1,:) - dxi(1,root_node);
        dxi(2,:) = dxi(2,:) - dxi(2,root_node);
    end

    %% Plotting the connected graph
    if(plot_dynamic_graph == 1)
        set(0,'CurrentFigure',figure_graph);
        plot(G); title('Dynamic Network Topology');
        xlim([-3 3]);
        ylim([-3 3]);
    end

    %% Normalizing the velocity limits
    if(normalize_velocities == 1)
        for i = 1:N
            if norm(dxi(:,i)) > dxmax
                dxi(:,i) = dxi(:,i)/norm(dxi(:,i))*dxmax;
            end
        end
    end
    %% Utilize barrier certificates for inter-robot
collision avoidance
    if(collision_avoidance == 1)
        dxi = si_barrier_cert(dxi, x);
    end

    %% Transform the single-integrator to unicycle dynamics
using the transformation we created earlier
    dxu = si_to_uni_dyn(dxi, x);

    %% Send/Set velocities to agents
    r.set_velocities(1:N, dxu);
    r.step(); % This function must be called to properly
set the velocities and execute them

    %% Performance evaluation metrics
    % Calculate the distance travelled in each iteration
    for i=1:N
```

```
        distance_travelled(i) = distance_travelled(i) +
norm(xi(:,i)-previous_xi(:,i));
    end
    previous_xi = xi;

    %if (stop_condition == 1 && length(leaves) == N-1)
    if ((stop_condition == 1) &&
(iteration_at_stopcondition == 0))
        fprintf('Stop condition (all inter-robot distances
within 0.5m) reached at %d',t);
        iteration_at_stopcondition = t;
    end

    %if((energy(t) <= desired_energy) &&
(iteration_at_minenergy == 0))
        %fprintf('Minimum energy condition (E<=10) reached
at %d',t);
    %end
    %% Movie making
    if(make_movie == 1)
        set(0,'CurrentFigure',r.figure_handle);
        Movieframe(t) = getframe(gcf);
    end

    %% Deleting unncessary figure/text handles
    if(plot_robot_index == 1)
        delete(fig_index); % delete the text objects (robot
indices) on the Robotarium figure
    end
    delete(fig_iter);


end

% Closing the Robotarium object properly
%r.debug();

%% Movie making
if(make_movie == 1)
    v =
VideoWriter('robotarium_simulation_expoeriment.mp4','MPEG-
4');
    open(v);
    writeVideo(v,Movieframe);
```

```
    close(v);
end

%% Saving the final figure if needed
%print(figure_robotarium,'RobotariumFigure','-depsc');
```

Main Program

```
clc;clear all;close all;
%% Algorithms
%% Choose your Rendezvous algorithm
% Newly Proposed Weighted Bearing Controllers
algorithm1 =
'weighted_bearing_consensus_using_RSS_and_DOA'; % It uses
the DOA of RSS and the RSS form wireless nework
measurements as control inputs
%algorithm =
'weighted_bearing_consensus_using_Range_and_Bearings'; % It
uses both range and bearing measurements from any sensors
as control inputs

% Coordinates based consensus (Rendezvous) algorithms -
using relative position measurements
%algorithm = 'coordinates_based_rendezvous' ; % (Baseline)
It relies on the full coordinates (relative positions) of
neighbor robots
algorithm2 =
'coordinates_based_connectivity_preserving_rendezvous' ; %
It is similar to the above but uses weights (artificial
potential fields)
%algorithm = 'coordinates_based_rendezvous_circumcenter' ;
% It relies on the circumcenter of all coordinates
(relative positions) of neighbor robots


% State of the Art (SOTA) Bearing-only
consensus(Rendezvous) algorithms
%algorithm = 'bearing_only_rendezvous_using_all_bearings';
% It uses bearing measurements of all neighbor robots from
each robot
%algorithm =
'bearing_only_rendezvous_using_min_and_max_bearings'; % It
uses only the min and max bearings of neighbors from each
robot
algorithm3 =
'bearing_only_rendezvous_using_enclosing_circles'; % It
uses bearing measurmeents and enclosing circles algorithm

% Other possible consensus controllers - both coordinates
and bearings based controllers
```

```
algorithm4 =
'bearing_only_rendezvous_using_average_bearing'; % It uses
mean of neighbor bearings from each robot
%algorithm =
'coordinates_based_rendezvous_with_mean_velocity'; % It
uses mean of all neighbor directions
%algorithm =
'coordinates_based_rendezvous_with_max_velocity'; % It uses
the coordinate of farthest neighbor
%algorithm =
'coordinates_based_rendezvous_with_min_velocity'; % It uses
th
%% Implementation
N=15;
t = linspace(1,1000,1000);
RSS_DOA_energy = rendezvous_experiments(N,algorithm1,1000);
pause(10);
coord__based_energy =
rendezvous_experiments(N,algorithm2,1000);
pause(10);
bearing_only_energy =
rendezvous_experiments(N,algorithm3,1000);
figure();
grid on;
hold on;
plot(t,RSS_DOA_energy);
plot(t,coord__based_energy);
plot(t,bearing_only_energy);
legend('RSS-DOA','connectivity preserving','bearing only');
```