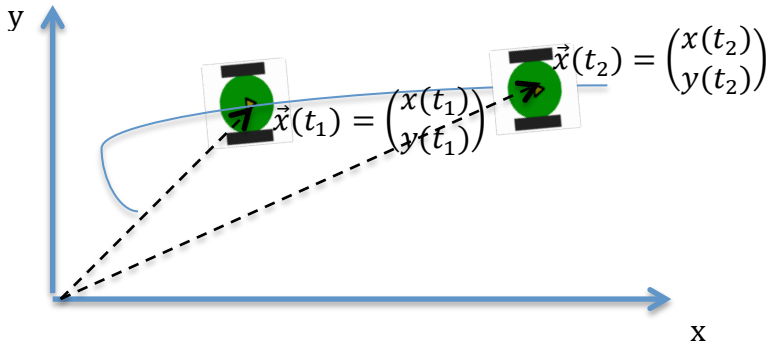
Week 1: Assignment 1

Prof. Solmaz Kia, solmaz@uci.edu
Kia Cooperative Systems Lab
Mechanical and Aerospace Engineering Department, University of California Irvine
July 10-15, 2017

Kinematics: the branch of mechanics concerned with the motion of objects without reference to the forces that cause the motion. In Kinematics, we characterize the relationship between displacement, velocity and acceleration of the objects.

In the following we assume that our object of interest moves in a 2D space (on a flat surface).

To keep track of displacement of an object on a 2D plane in a Cartesian coordinate system we need a reference point and 2 reference directions.



- Displacement then is $\Delta\vec{x} = \vec{x}(t_2) - \vec{x}(t_1)$
- Velocity: rate of change of position in time.
    - Consider a robot moving on a straight line with a constant velocity v. We can compute the displacement of this robot between time t_1 and time t_2 from

$$\Delta x = x(t_2) - x(t_1) = v(t_2 - t_1)$$

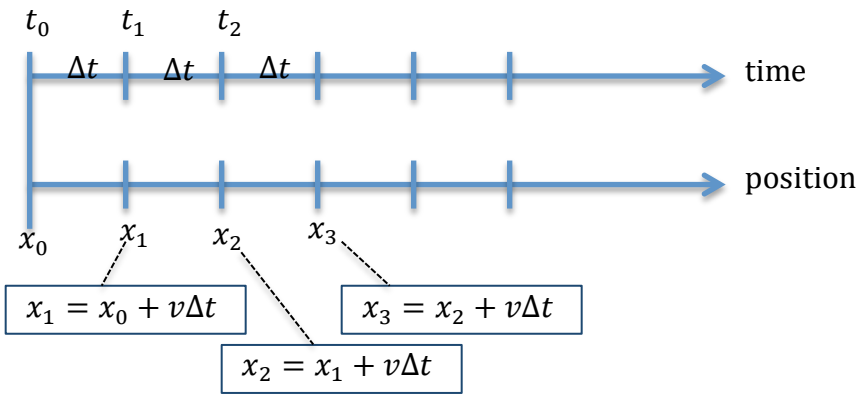
Starting from an initial position at time $t = 0$, the position of our robot at the consecutive times t is given by

$$x(t) = vt + x(0)$$

The equation above gives you the position at each time t. Lets say we are only interested in knowing the position at particular time $t_{i+1} =$

$t_i + \Delta t, i = 1,2,3,4, \ldots$, where $t_0 = 0$, and Δt is the sampling time (you can think of a radar which only samples the motion at particular times). Then, the position $x_{i+1} = x(t_{i+1})$ of our robot at each t_i is given by

$$x_{i+1} = x(t_{i+1}) = x_i + v\Delta t, \qquad \text{i=1,2,3,....}$$



Numerical example: Let $x(0) = 2\ m\ and\ v = 1.5\frac{m}{sec}$. You can use the following Matlab commands to visualize the motion of the robot for $0 \leq t \leq 3\ sec$:

```
close all;
x_0=2;
v=1.5;
delta_t=0.1;
for t=0:delta_t:3
    x_t=v*t+x_0;
    plot(t,x_t,'r+')
    hold on
end
```

```
close all;
x_0=2;
v=1.5;
delta_t=0.1;
t=(0:delta_t:3);
x_t=v*t+x_0;
plot(t,x_t,'rx')
```

```
close all;
x_0=2;
v=1.5;
delta_t=0.1;
t=(0: delta_t:3);
l_t=length(t);
x(1)=x_0;
for i=1:l_t-1
    x(i+1)=v*delta_t+x(i);
end
plot(t,x,'ro')
```

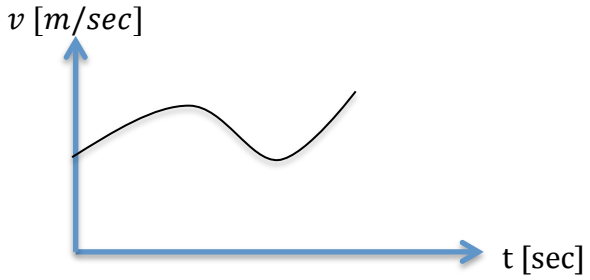
Add animation to your plots.

- Consider a robot now moving on a flat surface with a constant speed v_x in x direction and constant speed v_y in y direction. Then we can compute the displacement of the object between time t_1 and time t_2 from

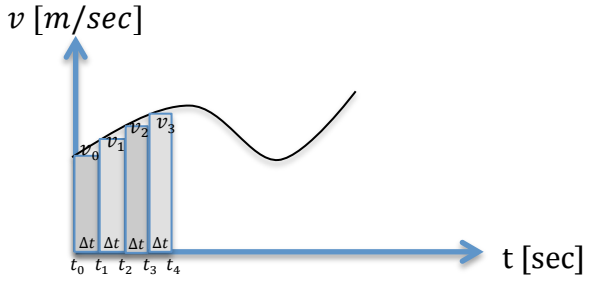
$$\Delta\vec{x} = \begin{pmatrix} x(t_2) - x(t_1) \\ y(t_2) - y(t_1) \end{pmatrix} = \begin{pmatrix} v_x(t_2 - t_1) \\ v_y(t_2 - t_1) \end{pmatrix}$$

Assignment: Can you describe the location of this robot at each time t_i? Let x(0)=2 and y(0)=2, and $t_0 = 0$. Let the constant speed at x direction be $v_x = 1\frac{m}{sec}$ and constant speed at y direction be $v_y = \frac{2m}{sec}$. Plot y(t) vs. x(t) for $0 \leq t \leq 5\ sec$.

- Now consider a robot moving on a straight line with a variable velocity v(t),



In this case the position of the robot at each time t is not given by $x(t) = vt + x(0)$. To compute the position we can use the following approximate approach. The idea is to discretize the time at a sample Δt and for each time $t_i \leq t \leq t_{i+1}$, we assume that the velocity of the robot is constant $v(t) = v_i = v(t_i)$.



Using this approach the position of the robot can be obtained from

$$x_{i+1} = x(t_{i+1}) \approx x_i + v(t_i)\Delta t, \qquad i = 1,2,3, \ldots$$

As you can expect, the smaller the size of Δt the more accurate your position estimate will be.

Numerical example: Let $x(0) = 2\ m\ and\ v(t) = 1 + sin(t)\frac{m}{sec}$. The exact position of the robot at each time t is given by

$$x(t) = t - \cos(t) + 1 + x(0), \qquad t \geq 0$$

Let is discretize time with a sampling period $\Delta t = 0.1\ sec$. Then, exact position of each robot is given by

$$x_i = x(t_i) = t_i - \cos(t_i) + 3$$

We can also approximate the location of the robots using the method described above:

$$x_{i+1} = x(t_{i+1}) \approx x_i + (1 + \sin(t_i))\Delta t, \qquad i = 1,2,3, \ldots$$

We can use the Matlab commands below to compare the accuracy of our approximate position calculation for $0 \leq t \leq 3\ sec$:

```
close all;
x_0=2;
delta_t=0.1;
t=(0: delta_t:3);
l_t=length(t);
x_approximate(1)=x_0;
x_exact(1) =x_0;
for i=1:l_t-1
   v(i)=1+sin(t(i));
   x_approximate(i+1)=x_approximate (i)+v(i)*delta_t;
   x_exact(i+1)=t(i+1)-cos(t(i+1))+1+x_0;
end
plot(t, x_approximate,'+b')
hold on
plot(t, x_exact,'*r')
xlabel('t')
ylabel('x')
```

Run your Matlab code with different sampling time size and see how size of Δt affects the accuracy of your position approximation. Add animation to your plots.

- Consider a robot now moving on a flat surface with a constant speed $v_x = 2\frac{m}{sec}$ in x direction and variable speed $v_y(t) = 1 + \sin(t)$. Starting from x(0)=2 m and y(0)=0, the exact location of the robot at each time t is given by

$$x(t) = x(0) + v_x t = 2 + 2t,$$
$$y(t) = y(0) + t - \cos(t) + 1 = t - \cos(t) + 1$$

Assignment: Can you describe the location of this robot at each time t_i using the approximate method described above?

Plot y(t_i) vs. x(t_i) for $0 \leq t \leq 5\,\text{sec}$ for both exact and approximate methods. Run your Matlab code with different sampling time size and see how size of Δt affects the accuracy of your position approximation. Add animation to your plots.

Week 2

Prof. Solmaz Kia, solmaz@uci.edu
Kia Cooperative Systems Lab
Mechanical and Aerospace Engineering Department, University of California Irvine
July 17-21, 2017

Graph Theory is a branch of mathematics, which studies the graphs. A graph is made up of vertices, nodes, or points, which are connected, by edges, arcs, or lines. A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be directed from one vertex to another.

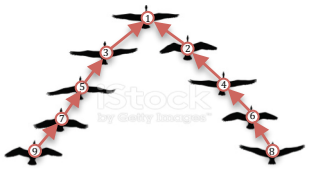


Fig. 1. A *directed graph* used to depict the reference tracking relation among a flock of geese (assumption is that each bird only focuses on the bird ahead of it) .

Fig. 2. An *undirected graph* used to depict the neighboring relation between southern California counties.

Application: graphs are used to model pairwise relations between objects. Graphs provide natural abstractions for how information is shared between agents in a network. Concepts such as neighbor sets, adjacency matrix, paths, etc in graph theory provide the language and the tools we need to design and program algorithms for multi-agent systems.

Objective: the objective of this week's reading and programing assignments is familiarize you with some basic concepts in graph theory and some of the applications of graph theory. Next week, you will use these concepts to design and simulate formation and coordination algorithms for a group of robots.

Example of use of graph theory:
Consider the map of the southern California. Our objective is to color this map such that no neighboring county has the same color. One solution is using the following graph theoretic approach:
Consider the graph representation of the neighboring counties given in Fig. 2. Let $V = \{1,2,3,4,5,6,7,8\}$ be the set of nodes in the graph. Let N_i represent the neighbor set of agent i:

$$N_1 = \{2\},\ N_2 = \{1,3\}, N_3 = \{2,4,5\}, N_4 = \{3,5,6\}, N_5 = \{3,4,6,7\}, N_6 = \{4,5,7,8\},$$
$$N_7 = \{5,6,8\}, N_8 = \{6,7\}$$

Let set of the available colors be color={red,blue,green,yellow,brown}. We implement the following procedure.
Let V_r, V_b, V_g, V_y be the set of red, blue, green and yellow nodes, respectively. Let N_r, N_b, N_g, N_y be the set of nodes that are neighbor of red, blue, green and yellow

nodes, respectively. Let V^{un-c}be the set of agents that are not colored yet.

Initialization:

$$V_r = \{\quad\}, V_b = \{\quad\}, V_g = \{\quad\}, V_y = \{\quad\}$$
$$N_r = \{\quad\}, N_b = \{\quad\}, N_g = \{\quad\}, N_y = \{\quad\}$$
$$V^{un-c} = V = \{1,2,3,4,5,6,7,8\}$$

1: set i=1, set color to red

2: color node i red:

$$V_r = \{1\}, V_b = \{\quad\}, V_g = \{\quad\}, V_y = \{\quad\}$$
$$N_r = N_r \cup N_i = \{2\}, N_b = \{\quad\}, N_g = \{\quad\}, N_y = \{\quad\}$$
$$V^{un-c} = V^{un-c} - \{i\} = \{2,3,4,5,6,7,8\}$$

3:choose the next node i as the smallest index in $V^{un-c} - N_r = \{3,4,5,6,7,8\}$, that is i=3

4: color node i red:

$$V_r = \{1,3\}, V_b = \{\quad\}, V_g = \{\quad\}, V_y = \{\quad\}$$
$$N_r = N_r \cup N_i = \{2,4,5\}, N_b = \{\quad\}, N_g = \{\quad\}, N_y = \{\quad\}$$
$$V^{un-c} = V^{un-c} - \{i\} = \{2,4,5,6,7,8\}$$

5: choose the next node i as the smallest index in $V^{un-c} - N_r = \{6,7,8\}$, that is i=6

6: color node i red:

$$V_r = \{1,3,6\}, V_b = \{\quad\}, V_g = \{\quad\}, V_y = \{\quad\}$$
$$N_r = N_r \cup N_i = \{2,4,5,7,8\}, N_b = \{\quad\}, N_g = \{\quad\}, N_y = \{\quad\}$$
$$V^{un-c} = V^{un-c} - \{i\} = \{2,4,5,7,8\}$$

7: since $V^{un-c} - N_r = \{\}$, switch the color to blue

8: choose the next node i as the smallest index in $V^{un-c} - N_b = \{2,4,5,7,8\}$, that is i=2

9: color node i blue:

$$V_r = \{1,3,6\}, V_b = \{2\}, V_g = \{\quad\}, V_y = \{\quad\}$$
$$N_r = \{2,4,5,7,8\}, N_b = N_b \cup N_i = \{1,3\}, N_g = \{\quad\}, N_y = \{\quad\}$$
$$V^{un-c} = V^{un-c} - \{i\} = \{4,5,7,8\}$$

10: choose the next node i as the smallest index in $V^{un-c} - N_b = \{4,5,7,8\}$, that is i=4

11: color node i blue:

$$V_r = \{1,3,6\}, V_b = \{2,4\}, V_g = \{\quad\}, V_y = \{\quad\}$$
$$N_r = \{2,4,5,7,8\}, N_b = N_b \cup N_i = \{1,3,5,6\}, N_g = \{\quad\}, N_y = \{\quad\}$$
$$V^{un-c} = V^{un-c} - \{i\} = \{5,7,8\}$$

12: choose the next node i as the smallest index in $V^{un-c} - N_b = \{7,8\}$, that is i=7

13: color node i green:

$$V_r = \{1,3,6\}, V_b = \{2,4,7\}, V_g = \{\quad\}, V_y = \{\quad\}$$
$$N_r = \{2,4,5,7,8\}, N_b = N_b \cup N_i = \{1,3,5,6,8\}, N_g = \{\quad\}, N_y = \{\quad\}$$
$$V^{un-c} = V^{un-c} - \{i\} = \{5,8\}$$

14: since $V^{un-c} - N_b = \{\}$, switch the color to green

15: choose the next node i as the smallest index in $V^{un-c} - N_g = \{5,8\}$, that is i=5

16: color node i green:

$$V_r = \{1,3,6\}, V_b = \{2,4,7\}, V_g = \{5\}, V_y = \{\quad\}$$
$$N_r = \{2,4,5,7,8\}, N_b = N_b \cup N_i = \{1,3,5,6,8\}, N_g = \{3,4,6,7\}, N_y = \{\quad\}$$
$$V^{un-c} = V^{un-c} - \{i\} = \{8\}$$

17: choose the next node i as the smallest index in $V^{un-c} - N_g = \{8\}$, that is i=8

18: color node i green:

$$V_r = \{1,3,6\}, V_b = \{2,4,7\}, V_g = \{5,8\}, V_y = \{\ \ \}$$
$$N_r = \{2,4,5,7,8\}, N_b = N_b \cup N_i = \{1,3,5,6,8\}, N_g = \{3,4,6,7\}, N_y = \{\ \ \}$$
$$V^{un-c} = V^{un-c} - \{i\} = \{\}$$

19: stop the process because $V^{un-c} = \{\}$, that is, no node needs coloring.

Graph coloring problem: the problem is, given m colors, find a way of **coloring** the vertices of a **graph** such that no two adjacent vertices are colored using same color.

Assignment: Google graph coloring and investigate how given a graph you can determine the minimum number of colors needed.

The following Matlab code preforms the graph coloring for the southern California map based on the algorithm described above.

Assignment: Assign coordinates to the nodes on a Cartesian coordinate and plot the nodes based on the colors assigned by the program above.

For example if node (1) is at (0,0) and node (2) is at (1,-1), then you can plot these nodes based on

```
X=cell(8);
X{1}=[0,0];
X{2}=[1,-1];
X{3}=…;
X{4}=…;
X{5}=…;
X{6}=…;
X{7}=…;
X{8}=…;
figure
hold on
for i=1:8
if any(i==V_r)
plot(cell2mat(X(i)),'or')
elseif any(i==V_b)
plot(cell2mat(X(i)),'ob')
elseif any(i==V_g)
plot(cell2mat(X(i)),'og')
elseif any(i==V_y)
plot(cell2mat(X(i)),'oy')
end
end
```

Connect the nodes to visually represent the graph. Try this algorithm on a different map. Can you think of an alternative algorithm which finishes in less that 19 steps?

```
close all
%definiting the graph
V=[1,2,3,4,5,6,7,8]; %node set
N=cell(8,1);
N{1}=[2];
N{2}=[1,3];
N{3}=[2,4,5];
N{4}=[3,5,6];
N{5}=[3,4,6,7];
N{6}=[4,5,7,8];
N{7}=[5,6,8];
N{8}=[6,7];
%colors
colors=['r','b','g','y'];
%initilization
N_r=[];
N_b=[];
N_g=[];
N_y=[];
V_r=[];
V_b=[];
V_g=[];
V_y=[];
V_un_c=V;
col='r';
N_c=N_r;
V_c=V_r;
Flag=1; %flag is used to detect when V_un_c is empty
        % 1: V_un_c not empty; 0: V_un_c is empty
while Flag %loop until V_un_c is empty
    if isempty(setdiff(V_un_c,N_c))
        if col=='r'
            col='b';
            N_r=N_c;
            V_r=V_c;
            clear N_c;
            clear V_c;
            N_c=N_b;
            V_c=V_b;
            if isempty(V_un_c)
                Flag=0;
            end
        elseif col=='b'
            col='g';
            N_b=N_c;
            V_b=V_c;
            clear N_c;
            clear V_c;
            N_c=N_g;
            V_c=V_g;
            if isempty(V_un_c)
                Flag=0;
            end
        else
            col='y';
            N_g=N_c;
            V_g=V_c;
            if isempty(V_un_c)
                Flag=0;
            end
        end
    else
    nod=min(setdiff(V_un_c,N_c));
    V_c=[V_c nod];
    N_c=[N_c cell2mat(N(nod))];
    V_un_c=setdiff(V_un_c,[nod]);
    end
    if col=='y'
        N_y=N_c;
        V_y=V_c;
    end
end
```

Point of Contact for this assignment

Yi-Fan Chung
yfchung@uci.edu
Ph.D. student
Kia Cooperative Systems Lab
Mechanical and Aerospace Engineering
University of California Irvine

Appendix

Union of sets: In set theory, the **union** (denoted by ∪) of a collection of sets is the set of all elements in the collection. Example:

$$A = \{1,4\ 5,9\}, B = \{3,4,5,6\},\ \ A \cup B = \{1,3,4,5,6,9\}.$$

Intersection of sets: In set theory, the **intersection** (denoted by ∩) of two sets *A* and *B* (*A* ∩ *B*) is the set that contains all elements of *A* that also belong to *B* (or equivalently, all elements of *B* that also belong to *A*), but no other elements.

$$A = \{1,4\ 5,9\}, B = \{3,4,5,6\},\ \ A \cap B = \{4,5\}.$$

Difference of sets: In set theory **set-theoretic difference** of set A and set B (denoted by A-B), is the set of elements in A that are not in B:

$$A = \{1,4\ 5,9\}, B = \{3,4,5,6\},\ \ A - B = \{1,9\}.$$

Week 3

Prof. Solmaz Kia, solmaz@uci.edu
Kia Cooperative Systems Lab
Mechanical and Aerospace Engineering Department, University of California Irvine
July 24-29, 2017

Technological advances in ad-hoc networking and miniaturization of electro-mechanical systems are making possible the use of large numbers of mobile agents (e.g., mobile robots, unmanned vehicles) to perform surveillance, search and rescue, transport and delivery tasks in aerial, underwater, space, and land environments. In all of these tasks the ideal operation is to take humans out of the operation loop and leave it to the mobile agents to figure out their actions autonomously to enable the group behavior. Enabling mobile agents to decide locally based on their own local information and information they can obtain from their neighboring agents (agents that can communicate information to them) to preform a global task is the subject of study in *distributed algorithm design* for mobile agents.

Focus and Objective: the focus of this week is on how to use graph theoretic tools along with laws of motion to devise algorithms for robot coordination problems. The particular problems we are going to focus on is the robot rendezvous problem (starting from random initial locations, the robots should eventually meet at one location by only knowing the location of their neighbors) and leader follower problem (starting from random locations we want all the agents follow a leader robot by only knowing the location of their neighbors).

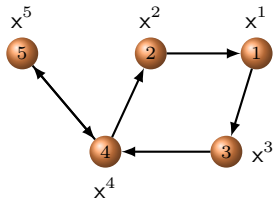
The first practice assignment for this week is designed to demonstrate the concepts that are essential in solving our problems of interest. In any coordination problem, having a path from each agent to all the other agents in the network plays a crucial role. For example, you can imagine that in a leader follower problem, every agent needs to have direct or indirect (through its neighbors) to the information of the leader. Graph connectivity captures is captured through identifying paths and trees in a graph.

Point of Contact for this assignment

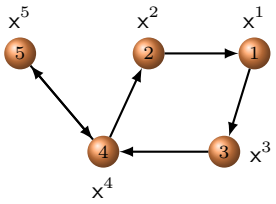
Hossein Moradian
hmoradia@uci.edu
Ph.D. student
Kia Cooperative Systems Lab
Mechanical and Aerospace Engineering
University of California Irvine

Consider some robots interacting according to the following network. An arrow from an agent to another indicates that the agent at the tip of the arrow can obtain information from the agent at the head of the arrow (you can imagine that the arrow indicate the agent at the tip sees the agent at the head).

Topology (1)



Topology (2)



1. Determine the adjacency matrix for both graphs.

2. Determine all the paths with two edges for graph (1)

3. Find a spanning tree in both graphs (If exists).

4. For each agent $i \in \{1, 2, 3, 4, 5\}$, determine the in-neighbors (agents that can receive information from agent i) and out-neighbors (agents that can send information to agent i).

5. What is the definition of a strongly connected directed graph (strongly connected digraph). Is any of the graphs strongly connected?

6. Agent 1 has a token which passes to one of its in-neighbors. Then the receiving agent passes the token to one of its in-neighbors. And this process repeats. Determine all the agents that can receive the token.

7. Do you think if it is possible to design a distributed leader follower algorithm for both of these typologies with agent 1 as leader (other agents in the network are expected to imitate the behavior of the leader.)

8. The communication and measurement devices have limit ranges. Figure 1 depicts a group of 6 robots along with their communication and measurement ranges.

   - Communication graph: draw a graph that captures the communication topology, that is, who can communicate with who (e.g., because agent 2 is in the communication range of agent 1, in your plot you should have an arrow from agent 2 to agent 1 to indicate that agent 2 can receive information from agent 1).
   - Sensing graph: draw a graph that captures the sensing topology, that is who can take measurement from who (e.g., because agent 3 is in the sensing zone of agent 6, in your plot you should have an arrow from agent 6 to agent 3 to indicate that agent 6 can obtain information from agent 3).

9. Let Topology (1) be the sensing graph of a team of 5 robots. Draw a plot similar to the one in Fig. 1 with sensing zones highlighted.

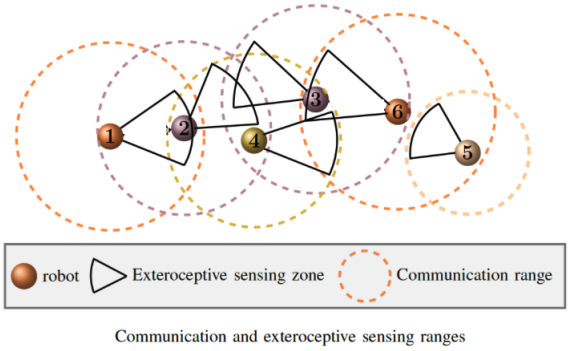


Figure 1: Communication and measurement range of mobile robots

10. Let Topology (2) be the communication graph of a team of 5 robots. Draw a plot similar to the one in Fig. 1 with communication circles highlighted.

The End.

Week 4

Prof. Solmaz Kia, solmaz@uci.edu
Kia Cooperative Systems Lab
Mechanical and Aerospace Engineering Department, University of California Irvine

July 31st-August 4th, 2017

The objective of this week's handout is to provide you supporting material so you can devise a distributed rendezvous algorithm for a group of mobile robots moving on a 2 dimensional (2D) flat surface.

Case of two robots: Consider two robots located on a straight line at $x^{(1)}(0) = 0,$ and robot 2 starts at $x^{(2)}(0) = 5$. Here, $x^{(1)}(t_i)$ is the location of robot (1) at time t_i, and $x^{(2)}(t_i)$ is the location of robot (2) at time t_i.

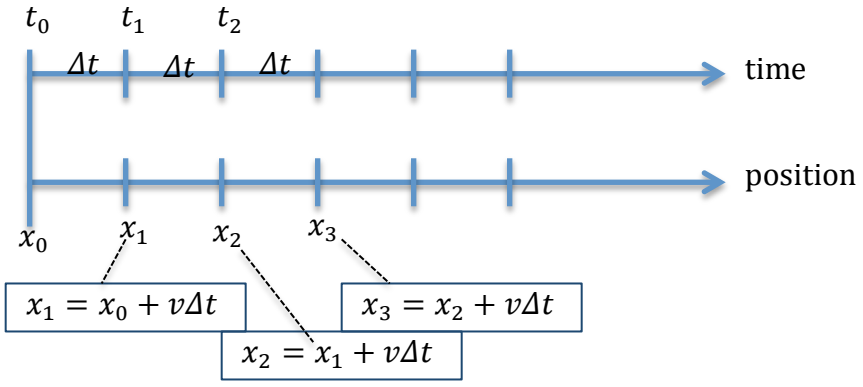
From the week 1's notes recall the following:

Starting from an initial position at time $t = 0$, the position of our robot at the consecutive times t is given by

$$x(t) = vt + x(0)$$

The equation above gives you the position at each time t.
Lets say we are only interested in knowing the position at particular time $t_{i+1} = t_i + \Delta t, i = 1,2,3,4, \ldots$, where $t_0 = 0$, and Δt is the sampling time (you can think of a radar which only samples the motion at particular times t_i). Then, the position $x_{i+1} = x(t_{i+1})$ of our robot at each t_i is given by

$$x_{i+1} = x(t_{i+1}) = x_i + v\Delta t, \qquad \text{i=0,1,2,....}$$



Let sampling time to be $\Delta t = 0.05\ sec$. Each robot has a sensor onboard that allows it to measure its relative distance from the other robot, that is, robot (2) can measure $x^{(1)}(t_i) - x^{(2)}(t_i)$ and robot (1) can measure $x^{(2)}(t_i) - x^{(1)}(t_i)$.

- Consider a case that robot (2) is stationary and Robot (1) wants to meet up with Robot (2), that is stating from $x^{(1)}(0) = 0$ Robot (1) wants to reach to $x^{(1)}(t_i) = x^{(2)}(0) = 5$ at some time $t_i > 0$ and stop there. A reasonable approach for Robot (1) is to adjust its velocity according to its distance measurement $x^{(2)}(t_i) - x^{(1)}(t_i)$,

$$v^{(1)}(t_i) = G(x^{(2)}(t_i) - x^{(1)}(t_i))$$

G is the adjustment gain. You can imagine that if you choose G too large robot (1) will go too fast and pass by robot (2), if you choose G too small it is going to take a long time for robot (1) to meet up with robot (2).
The equations of motion of the robots are

$$x^{(2)}{}_{i+1} = x^{(2)}{}_i, \qquad \text{robot (2) is stationary so its location does not change with time}$$

$$x^{(1)}{}_{i+1} = x^{(1)}{}_i + G(x^{(2)}(t_i) - x^{(1)}(t_i))\Delta t$$

  - Matlab file below simulates this meet up scenario. Change gain G and see how the response of robot (1) changes?
  - Can you add animation to this simulation showing robot (1) moving on a straight line towards the stationary robot (2).

```
clear
close all
x_1(1)=0;% initial location of robot 1
x_2(1)=5;% initial location of robot 2
gap=1;% the desired gap between robot 1 and 2
delta_t=0.05; %sampling time step
end_time=10;
time=0:delta_t:end_time;
length_time=length(time);
G=1;%

for i=1:length_time-1
    x_2(i+1)=x_2(i);
    x_1(i+1)=x_1(i)+G*(x_2(i)-x_1(i))*delta_t;
    i=i+1;
end

figure('Units','inches',...
'Position',[0 0 7.2 5],...
'PaperPositionMode','auto');

plot(time,x_1,'r',time,x_2,'b')
legend('robot 1','robot 2')
title('trajectory of two robots vs time')
ylabel('x')
xlabel('time')
axis([0 end_time -1 6])

figure('Units','inches',...
'Position',[8 0 7.2 5],...
'PaperPositionMode','auto');
plot(time,x_2-x_1)
title('distance between robot 1 and 2')
ylabel('x_2-x_1')
xlabel('time')
```

- Can you modify the code above for a 2D case? Lets say

$$\begin{pmatrix} x^{(2)}{}_{i+1} \\ y^{(2)}{}_{i+1} \end{pmatrix} = \begin{pmatrix} x^{(2)}{}_{i} \\ y^{(2)}{}_{i} \end{pmatrix},$$

robot (2) is stationary so its location does not change with time

$$\begin{pmatrix} x^{(1)}{}_{i+1} \\ y^{(1)}{}_{i+1} \end{pmatrix} = \begin{pmatrix} x^{(1)}{}_{i} + G(x^{(2)}(t_i) - x^{(1)}(t_i))\Delta t \\ y^{(1)}{}_{i} + G(y^{(2)}(t_i) - y^{(1)}(t_i))\Delta t \end{pmatrix}$$

Assume that the initial conditions are

$$\begin{pmatrix} x^{(1)}(0) \\ y^{(1)}(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} x^{(2)}(0) \\ y^{(2)}(0) \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

- Consider a case that the robots want to start from their initial conditions and meet up at somewhere and stop there, that is stating from $x^{(1)}(0) = 0$ and $x^{(2)}(0) = 5$ the two robots to meet up somewhere $x^{(1)}(t_i) = x^{(2)}(t_i)$ at some time $t_i > 0$ and stop there. A reasonable approach for Robot (1) is to adjust its velocity according to its distance measurement $x^{(2)}(t_i) - x^{(1)}(t_i)$,

$$v^{(1)}(t_i) = G_1(x^{(2)}(t_i) - x^{(1)}(t_i))$$

and also to robot (2) to do the same

$$v^{(2)}(t_i) = G_2(x^{(1)}(t_i) - x^{(2)}(t_i))$$

The equations of motion of the robots are

$$x^{(1)}{}_{i+1} = x^{(1)}{}_{i} + G_1\left(x^{(2)}(t_i) - x^{(1)}(t_i)\right)\Delta t,$$

$$x^{(2)}{}_{i+1} = x^{(2)}{}_{i} + G_2\left(x^{(1)}(t_i) - x^{(2)}(t_i)\right)\Delta t.$$

  - Matlab file below simulates this meet up scenario. Change gains $G_1 and\ G_2$ and see how the response of robots changes? What do you expect if you choose $G_1 = G_2$, $G_1 > G_2$ or $G_1 < G_2$?
  - Can you add animation to this simulation showing robot (1) moving on a straight line towards the stationary robot (2).
  - Can you modify the code above for a 2D case? Lets say

$$\begin{pmatrix} x^{(1)}{}_{i+1} \\ y^{(1)}{}_{i+1} \end{pmatrix} = \begin{pmatrix} x^{(1)}{}_{i} + G_{1x}(x^{(2)}(t_i) - x^{(1)}(t_i))\Delta t \\ y^{(1)}{}_{i} + G_{1y}(y^{(2)}(t_i) - y^{(1)}(t_i))\Delta t \end{pmatrix}$$

$$\begin{pmatrix} x^{(2)}{}_{i+1} \\ y^{(2)}{}_{i+1} \end{pmatrix} = \begin{pmatrix} x^{(2)}{}_{i} + G_{2x}(x^{(1)}(t_i) - x^{(2)}(t_i))\Delta t \\ y^{(2)}{}_{i} + G_{2y}(y^{(1)}(t_i) - y^{(2)}(t_i))\Delta t \end{pmatrix}$$

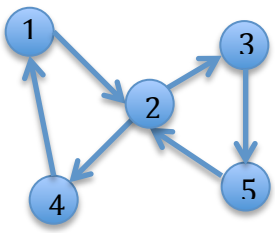
Assume that the initial conditions are

$$\begin{pmatrix} x^{(1)}(0) \\ y^{(1)}(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

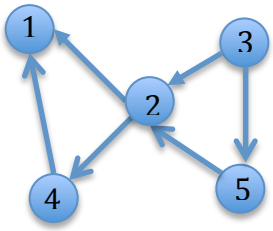
$$\begin{pmatrix} x^{(2)}(0) \\ y^{(2)}(0) \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

```
clear
close all
x_1(1)=0;% initial location of robot 1
x_2(1)=5;% initial location of robot 2
gap=1;% the desired gap between robot 1 and 2
delta_t=0.05; %sampling time step
end_time=10;
time=0:delta_t:end_time;
length_time=length(time);
G1=1;%
G2=1;
for i=1:length_time-1
    x_1(i+1)=x_1(i)+G1*(x_2(i)-x_1(i))*delta_t;
    x_2(i+1)=x_2(i)+G2*(x_1(i)-x_2(i))*delta_t;
    i=i+1;
end

figure('Units','inches',...
'Position',[0 0 7.2 5],...
'PaperPositionMode','auto');

plot(time,x_1,'r',time,x_2,'b')
legend('robot 1','robot 2')
title('trajectory of two robots vs time')
ylabel('x')
xlabel('time')
axis([0 end_time -1 6])

figure('Units','inches',...
'Position',[8 0 7.2 5],...
'PaperPositionMode','auto');
plot(time,x_2-x_1)
title('distance between robot 1 and 2')
ylabel('x_2-x_1')
xlabel('time')
```

- Now consider a case of 5 robots with two possible sensing topologies below. Here an arrow from robot k to robot j means that robot k can measure its relative pose from robot (j), that is robot (k) can obtain $x^{(j)}(t_i) - x^{(k)}(t_i)$ and $y^{(j)}(t_i) - y^{(k)}(t_i)$. Then, robot (k) can use this information to adjust its velocity. Our objective is for robots to start form the initial conditions given below and meets up at some point. The challenge here is that each robot can only take measurement from some subset of its team members.



Topology 1

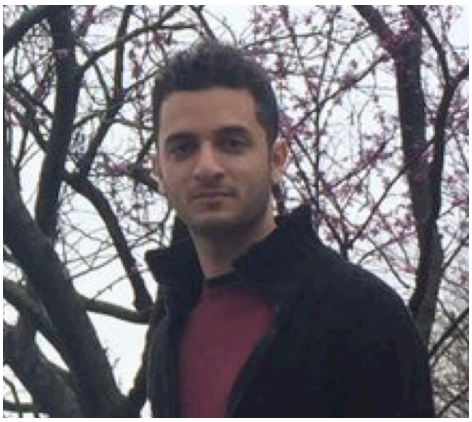


Topology 2

$$\begin{pmatrix} x^{(1)}(0) \\ y^{(1)}(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \end{pmatrix}, \begin{pmatrix} x^{(2)}(0) \\ y^{(2)}(0) \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}, \begin{pmatrix} x^{(3)}(0) \\ y^{(3)}(0) \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \end{pmatrix}, \begin{pmatrix} x^{(4)}(0) \\ y^{(4)}(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} x^{(5)}(0) \\ y^{(5)}(0) \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

- Write a rendezvous algorithm for these robots for each of the sensing topologies given above. Is there any relationship between the sensing topology and the rendezvous transient response you observe?

Point of Contact for this assignment



Navid Rezazadeh
nrezazad@uci.edu
M.Sc./Ph.D. student
Kia Cooperative Systems Lab
Mechanical and Aerospace Engineering
University of California Irvine