Decentralized traffic control in a cross junction- Two agent case using target point control and hybrid approach

Kumaraguru Sivasankaran[*†]

Abstract

Today, mobile vehicles on the roads follow traffic signals controlled by outdated technology. For example, a pile of cars tend to wait in a junction even though the other cross roads may be empty. This is due to fixed traffic control strategies which are without any real time optimization. With maturity in connected vehicle technologies viz V2V and V2I, it demands to review the existing traffic signal control algorithms. Although this problem has been studied in past, there are new variables in the system which makes the problem as new. For instance, in the world of connectivity, the user(driver) can inform the infrastructure on the start and end destination. This could help the overall network to know the traffic load information with high degree of precision. With advances in field of distributed optimization for large networks, decentralized control of traffic signals is possible ensuring minimum waiting time. In all, this will result in huge fuel savings, reduce the carbon footprint in the existing environment and improve the ride experience. This could be extended to a scenario of completely autonomous/ connected vehicles in future with no necessity of physical traffic signals and saving much on infrastructure costs.

Keywords: *Hybrid systems, Target point control, Multi-agents systems*

1. INTRODUCTION

The literature on multi-agent coordination strategies traditionally focuses on the design of localized coordination rules with provable, global properties such as achieving and maintaining formation, covering areas, or tracking boundaries. However, what is actually deployed on teams must also be safe in the sense that collisions are avoided. This calls for different coordination laws including collision avoidance. We can see it as a precursor towards having a decentralized approach applicable to a road traffic control problem. As the connectivity between robots increase, there is not much requirement to have always a physical sensor measuring the distance. Instead, a robot will be able to locate itself in a map and can see how far the other robots in the localized map. This is not far from achievable given the current advances in GPS and SLAM techniques. With this premise, the goal of this project work is to devise a coordinated control law to implement a collision avoidance strategy and at the same time minimize the waiting time of two agents at a cross junction. This idea has similarities to the work by Tomlin et al. (1998) for pairs of kinematic aircraft and based on global optimal control. In this project, a ground based road navigation problem is studied under the assumption that the agents are connected and able to share the information of their states to nearest agents. The outline of this report is as follows: In section 2 we briefly recall the target point control approach used for formation control and also the hybrid system control. In Section 3, we show how the decentralized control strategy is designed. In Section 4, the simulation results of the problem under study is discussed. In Section 5, the project work is summarized with concluding remarks and future directions.

2. BACKGROUND

2.1. Target point control

There is vast literature on multi-agent formations control where the problem is to maintain a specific formation by choosing a cooperative control input using the local sensed measurements about their neighbors. Due to the inherent problems posed by gradient control strategy, Mou et al. (2015) proposed an improved strategy called "target point control". It can be explained in simple terms as following: Consider a two agent system in a plane with one follower and one leader whose positions are denoted by x,y $\in R^2$ respectively. The target point in this case is defined as:

$$\tau(x,y) = \begin{cases} x + \frac{||y-x||-d}{||y-x||}(y-x), & \text{if } x \neq y \\ y + [d \;\; 0]', & \text{if x=y} \end{cases}$$

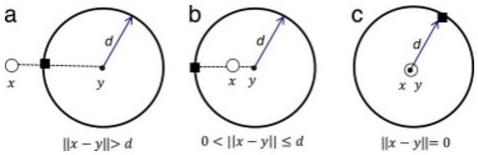


Figure 1: Target point control approach

*This work is a report submitted for the course project of AAE 590 - Distributed Network Control - Spring 2017

†[1]Kumaraguru Sivasankaran is a graduate student in department of Aeronautical and Astronautical Engineering, Purdue University, West Lafayette, Indiana, United States

2.2. Hybrid system control

Hybrid dynamical systems consist of a family of continuous or discrete-time subsystems and a rule that determines the switching between them. The switching behavior of these systems may be generated by the changing dynamics at different operating regions. Hybrid dynamical systems arise also when switching controllers are used to achieve stability and improve performance as shown in Figure 2. Typical examples of such systems are computer disk drives, constrained mechanical systems, switching power converters, and automotive power-train applications. Mathematically, such hybrid systems can be modeled by the equations:

$$\dot{x} = f(x(t), q(t), u(t))$$

$$q(t^+) = \delta(x(t), q(t))$$

where $x(t) \in R^n$ is the continuous state, $q(t) \in \{1,2,...,N\}$ is the discrete state that indexes the subsystems $f_{q(t)}, u(t)$ can be a continuous control input or an external (reference or disturbance) signal to the continuous part, and δ is the switching law that describes the logical and/or discrete event system dynamics.

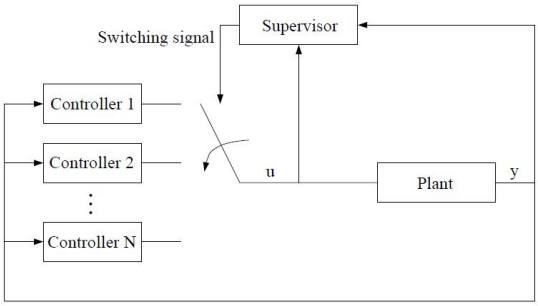


Figure 2: Switching controller feedback architecture

3. PROBLEM STATEMENT

3.1. Problem formulation

Consider two mobile cars trying to navigate through a cross road by avoiding the possibility of collision. The main focus on this problem is to establish a stopping rule as one car waits till the other car passes the intersection during the possibility of collision. Agents are modelled as having different dynamical capability using double integrator dynamics:

$$\begin{bmatrix} \dot{p}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} 0 & I_{2\times 2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_i \\ v_i \end{bmatrix} + \begin{bmatrix} 0 \\ I_{2\times 2} \end{bmatrix} u_i$$

where $p_i \in \mathbb{R}^2$,$v_i \in \mathbb{R}^2$,$u_i \in \mathbb{R}^2$ are the position, velocity and acceleration of the agent i respectively. Kindly note that the upper bound on acceleration and velocity has been fixed to an nominal value. Let the initial conditions be defined as in Table 1:

Table 1: Initial conditions and destination point

Agent	Initial position	Initial Velocity	Destination
Agent i	$\begin{bmatrix} p_{xi} \\ p_{yi} \end{bmatrix}$	$\begin{bmatrix} v_{xi} \\ v_{yi} \end{bmatrix}$	$\begin{bmatrix} r_{xi} \\ r_{yi} \end{bmatrix}$
Agent j	$\begin{bmatrix} p_{xj} \\ p_{yj} \end{bmatrix}$	$\begin{bmatrix} v_{xj} \\ v_{yj} \end{bmatrix}$	$\begin{bmatrix} r_{xj} \\ r_{yj} \end{bmatrix}$

A predefined controller which ensures the car to reach the destination point r_i would be

$$u_i = -k_1(p_i - r_i) - k_2 v_i$$

3.2. Problem approach

First we find the point of intersection of two agents at which collision might possibly occur. In the two agent case, we have the direction of motion of agent 1 and agent 2 as follows:

$$A_i = \begin{bmatrix} p_{xi} \\ p_{yi} \end{bmatrix} + \mu \begin{bmatrix} p_{xi} - r_{xi} \\ p_{yi} - r_{yi} \end{bmatrix} = \begin{bmatrix} p_{xi} \\ p_{yi} \end{bmatrix} + \mu \begin{bmatrix} v_{xi} \\ v_{yi} \end{bmatrix} ...(1)$$

$$A_j = \begin{bmatrix} p_{xj} \\ p_{yj} \end{bmatrix} + t \begin{bmatrix} p_{xj} - r_{xj} \\ p_{yj} - r_{yj} \end{bmatrix} = \begin{bmatrix} p_{xj} \\ p_{yj} \end{bmatrix} + t \begin{bmatrix} v_{xj} \\ v_{yj} \end{bmatrix} ...(2)$$

Solving equations (1) and (2) for the unknowns, we can get the coordinates where collision might happen provided the relative distance between the two agents is decreasing (ie. the agents are coming close towards each other). Here we consider that each agent know the information on current position and destination position/ velocity vector of other agent.

$$A_i = A_j = \begin{bmatrix} p_{xi} \\ p_{yi} \end{bmatrix} + \mu \begin{bmatrix} p_{xi} - r_{xi} \\ p_{yi} - r_{yi} \end{bmatrix} = \begin{bmatrix} p_{xj} \\ p_{yj} \end{bmatrix} + t \begin{bmatrix} p_{xj} - r_{xj} \\ p_{yj} - r_{yj} \end{bmatrix}$$

$$\mu \begin{bmatrix} p_{xi} - r_{xi} \\ p_{yi} - r_{yi} \end{bmatrix} - t \begin{bmatrix} p_{xj} - r_{xj} \\ p_{yj} - r_{yj} \end{bmatrix} = \begin{bmatrix} p_{xj} \\ p_{yj} \end{bmatrix} - \begin{bmatrix} p_{xi} \\ p_{yi} \end{bmatrix}$$

$$\begin{bmatrix} p_{xi} - r_{xi} & r_{xj} - p_{xj} \\ p_{yi} - r_{yi} & r_{yj} - p_{yj} \end{bmatrix} \begin{bmatrix} \mu \\ t \end{bmatrix} = \begin{bmatrix} p_{xj} - p_{xi} \\ p_{yj} - p_{yi} \end{bmatrix}$$

$$\begin{bmatrix} \mu \\ t \end{bmatrix} = \begin{bmatrix} p_{xi} - r_{xi} & r_{xj} - p_{xj} \\ p_{yi} - r_{yi} & r_{yj} - p_{yj} \end{bmatrix}^{-1} \begin{bmatrix} p_{xj} - p_{xi} \\ p_{yj} - p_{yi} \end{bmatrix}$$

$$\begin{bmatrix} \mu \\ t \end{bmatrix} = \begin{bmatrix} p_{xi} - r_{xi} & r_{xj} - p_{xj} \\ p_{yi} - r_{yi} & r_{yj} - p_{yj} \end{bmatrix}^{-1} \begin{bmatrix} p_{xj} - p_{xi} \\ p_{yj} - p_{yi} \end{bmatrix}$$

By substituting either μ in (1) or t in (2), we can get the point of intersection τ. We create two circles of diameter d_1 and d_2 around the point of intersection with the condition $d_1 > d_2$. Here, the circle with diameter d_2 is called safety circle and the circle with diameter d_1 is called circle of influence. The distance $|d1 - d2|$ will be the safe distance for braking.
Let τ be the point of intersection of the two agents. The target points are calculated as

$$d_{1i} = p_i + \frac{||\tau - p_i|| - d_1}{||\tau - p_i||}(\tau - p_i)$$

$$d_{2i} = p_i + \frac{||\tau - p_i|| - d_2}{||\tau - p_i||}(\tau - p_i)$$

$$d_{1j} = p_j + \frac{||\tau - p_j|| - d_1}{||\tau - p_j||}(\tau - p_j)$$

$$d_{2j} = p_j + \frac{||\tau - p_j|| - d_2}{||\tau - p_j||}(\tau - p_j)$$

These target points help to decide on the points where to switch the control strategy depending on relative velocity and relative position information.

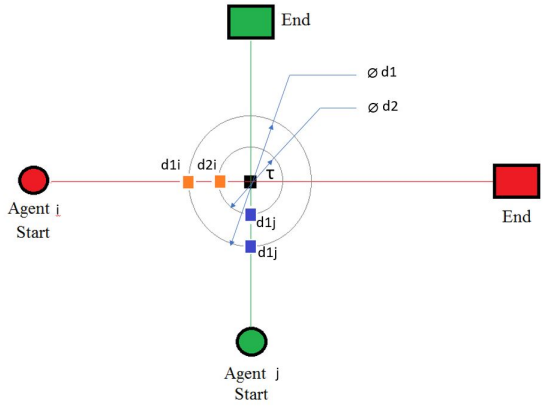


Figure 3: Two agents negotiating a cross junction - a target point control approach

3.3. Controller design

Case (a): When agent i enters the circle of influence ie. $|p_i - \tau| < d_1$ and if the agent j is outside the circle of influence ie. $|p_j - \tau| > d_1$, then, the agent i follows the pre-designed controller without any change. It stays in mode 1 of the hybrid control.

$$u_i = -k_1(p_i - r_i) - k_2 v_i$$

Case (b): When agent i enters the circle of influence ie. $|p_i - \tau| < d_1$ and if the agent j is already inside the circle of influence ie. $|p_j - \tau| \leq d_1$, then the agent i is supposed to stop at d_{2i} and hence, it will move into mode 2 of hybrid control:

$$u_i = -k_1(p_i - d_{2i}) - k_2 v_i$$

The agent i switches to pre-designed controller when the condition becomes $|p_j - \tau| \geq d_2$ or Δv_{ij} is increasing. Hence, the agent i will continue to go towards the desired destination.
The controller for agent j can also be designed in the same approach as explained in above two cases. Note that, this controller is distributed in the sense that its application only requires the other agent's position and destination or velocity information under the assumption that the values of distance of safety and distance of influence are predefined for all cars.

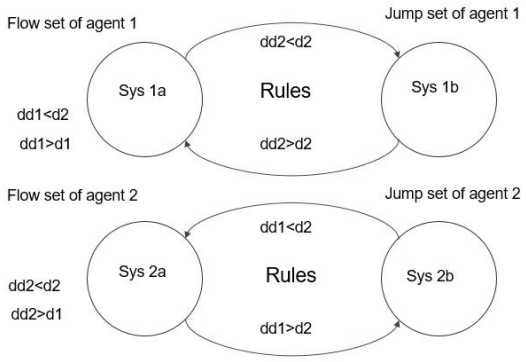


Figure 4: Switching controller feedback architecture

4. SIMULATION RESULTS

4.1. Methodology

Here, some of the basic terminologies in hybrid system is discussed for better understanding.
Flow set: The basis set of differential equations defining the system dynamics. This is where the system stays while the guard conditions are not violated.
Jump set: New set of differential equations defining the system dynamics. This set may necessarily not be limited to one.
Guard condition: The condition which is evaluated to define the system dynamics and also to help for transition if violated
Reset condition: The condition which brings back the system from jump set to flow set
As the system is described by double integrator dynamics and with switching controllers, solving using readily available solvers is not possible. Hence, a numerical approach to solving the system of differential equations is presented here. We have used a Runge-Kutta 4 method to solve this problem numerically. The steps involved for solving a single differential equation:

$$\dot{y} = f(t,y), y(t_0) = y_0$$

is as follows:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$t_{n+1} = t_n + h$$

for n = 1, 2, 3...

$$k_1 = f(t_n, y_n)$$

$$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$$

$$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$$

$$k_4 = f(t_n + h, y_n + hk_3)$$

The same has been extended to rest of the system equations as well. At every time step, the two sets of system dynamics' equation is solved and the guard conditions are

checked. If the guard conditions are violated for a particular flow set, the system dynamics get changed to the jump set. When the state reset conditions are met, the system comes back to original flow set. In our problem, the cooperative control is incorporated using guard condition which depends on the states of other agents. The complete MATLAB code is included in the appendix for reference.

4.2. Figures and Tables

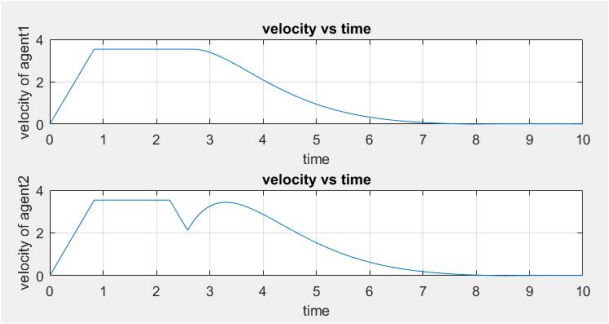


Figure 5: Velocity vs time plot

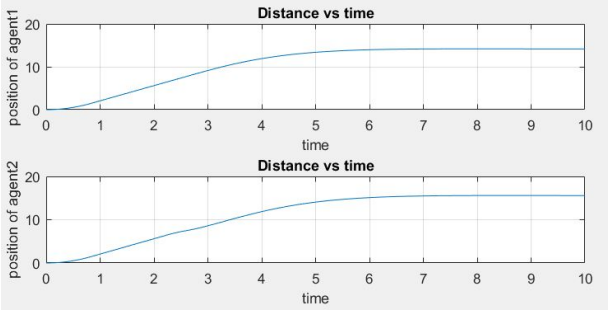


Figure 6: Position vs time plot

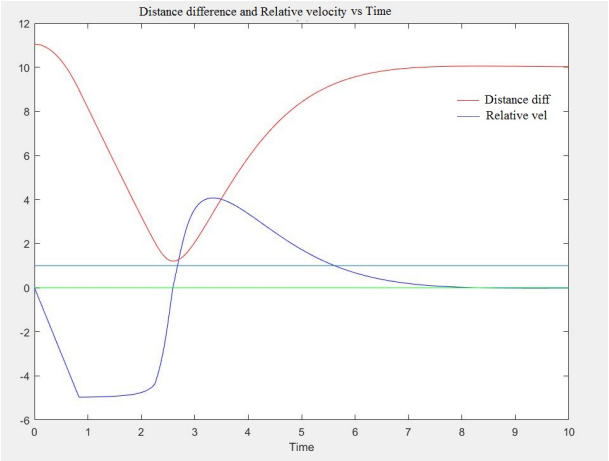


Figure 7: Distance difference, Relative velocity vs time plot

4.3. Results discussion

The figures in next page show the different positions of the two agents while crossing a junction at different time intervals. The bigger circle represents the influence circle and the smaller circle represents the safety circle. As we could relate with the various time plots, the slower agent reduces its velocity leaving way to the faster agent and then recovers speed without stopping at the junction while avoiding collision.

5. GENERAL FRAMEWORK

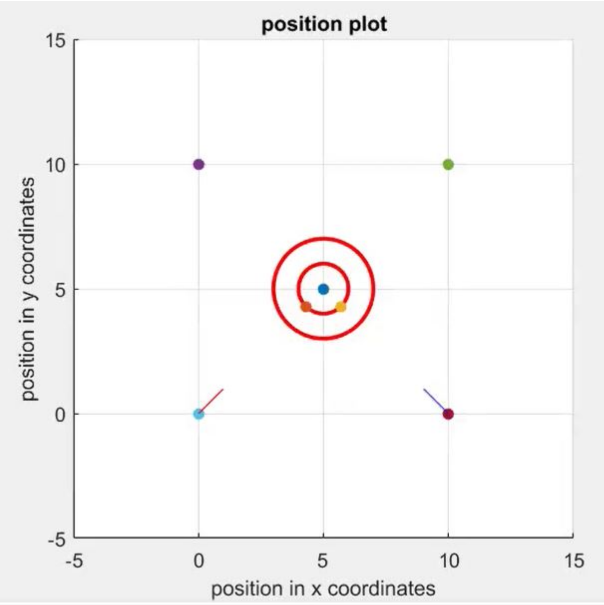
A limitation in this work is that there is no path pursuit algorithm and hence not immediately scalable. There are no constraints on the agent to follow a given path and the proposed controller works only for shorter distances between start and destination. Second, the target point is fixed in advance considering the path taken by the robot and considered to be not changing. This will not be the same in real case and a separate collision prediction algorithm would be needed to find the target points. To overcome these limitations, a general framework is proposed to include the following:

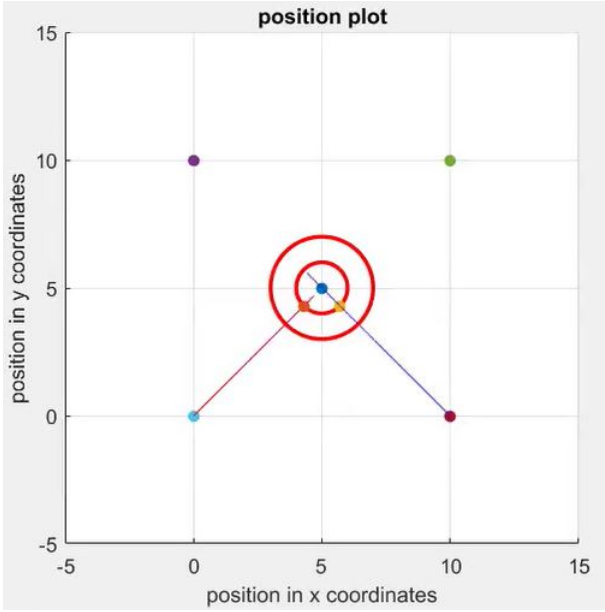
- To design a path following algorithm for double integrator dynamics allowing variable speed range similar to pure pursuit algorithm [3]. This will make the robot follow a defined path.
- To have a SLAM based navigation and GPS based navigation algorithm where the robot locates itself in the map and also knows the details about it's neighbours in a given circular area. It will also help to calculate the target point where a potential collision is about to happen.
- To create a set of switching controllers to speed up/down/stop for a given conflict situation similar to the protocols designed by TCAS system for air traffic control[5]. Here, the kinematic limitations of each agent can be included.
- To tackle congestion control, we can implement algorithm similar to AIMD to allocate lanes dynamically and give priority to lanes when there is saturating traffic flow.

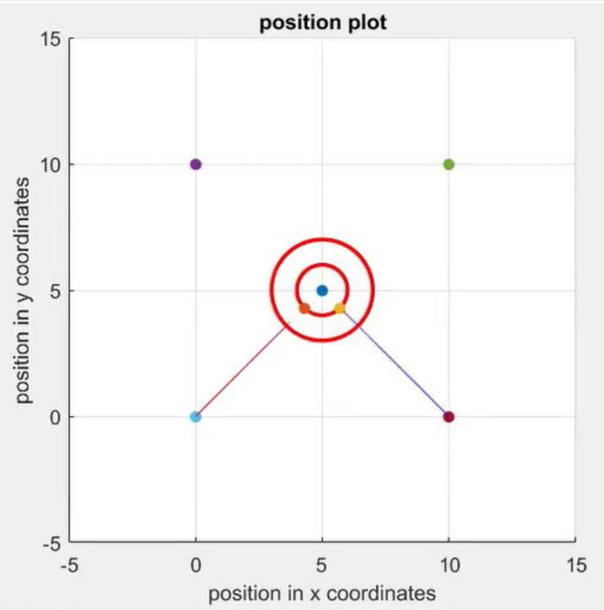
We believe this kind of framework when implemented in a multiagent network can bring more autonomy to the system and also improve the traffic conditions. Since this is beyond the scope of simulation, a physical implementation of the proposed idea can be pursued as future work.

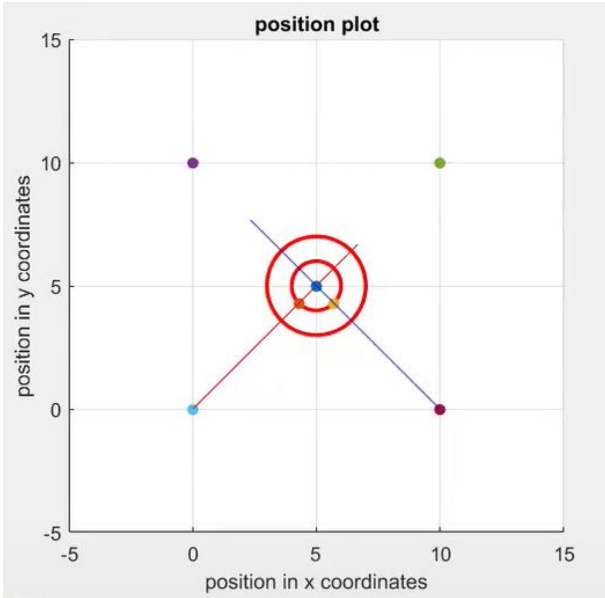
6. CONCLUSION

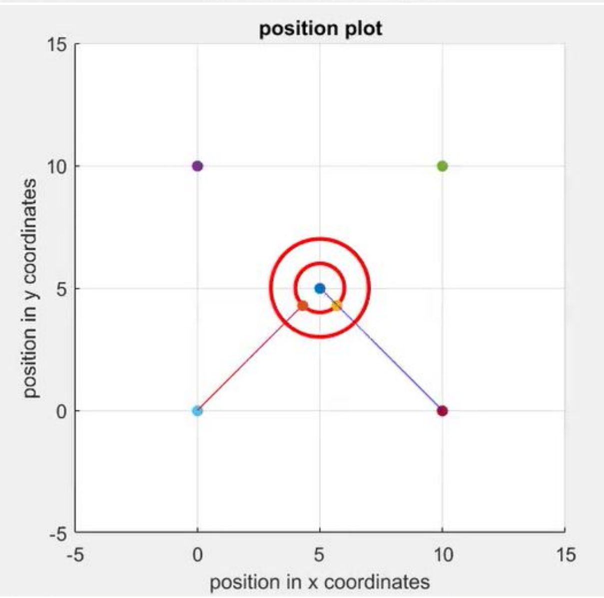
In this work, we have reviewed the hybrid system and target point control approach for devising a decentralized road traffic control. Then, we have showed a method to devise a decentralized controller for collision avoidance

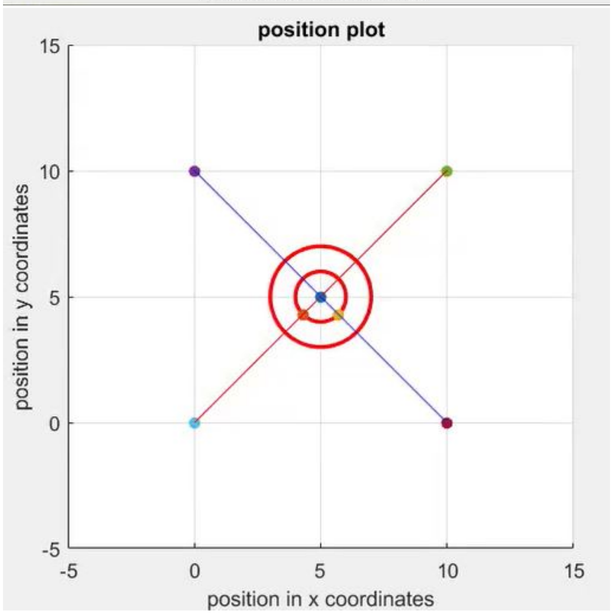

position plot
position in y coordinates
position in x coordinates
15
10
5
0
-5
-5
0
5
10
15


position plot
position in y coordinates
position in x coordinates
15
10
5
0
-5
-5
0
5
10
15


position plot
position in y coordinates
position in x coordinates
15
10
5
0
-5
-5
0
5
10
15


position plot
position in y coordinates
position in x coordinates
15
10
5
0
-5
-5
0
5
10
15


position plot
position in y coordinates
position in x coordinates
15
10
5
0
-5
-5
0
5
10
15


position plot
position in y coordinates
position in x coordinates
15
10
5
0
-5
-5
0
5
10
15

in a two agent case crossing a cross junction. A general framework is proposed for ground vehicle traffic control drawing similarities from a TCAS system in a decentralized approach.

APPENDIX

Appendix 1: MATLAB code for the simulation of two agent case discussed in this work.

ACKNOWLEDGMENT

I sincerely thank Professor Shaoshuai Mou for his constant guidance and support throughout this project work. I also thank Borrmann et al. for the inspiring paper [4] that led me to this work.

References

[1] S. Mou, M. Cao, A.S. Morse. Target-Point Formation Control. Automatica, 2015, 61, 113-118.

[2] Antsaklis, Panos J., Xenofon D. Koutsoukos, and N. Dame. "Hybrid systems control." Encyclopedia of Physical Science and Technology 7 (2002): 445-458.

[3] Coulter, R. Craig. Implementation of the pure pursuit path tracking algorithm. No. CMU-RI-TR-92-01. CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1992.

[4] Borrmann, Urs, et al. "Control barrier certificates for safe swarm behavior." IFAC-PapersOnLine 48.27 (2015): 68-73.

[5] Kuchar, J. E., and Ann C. Drumm. "The traffic alert and collision avoidance system." Lincoln Laboratory Journal 16.2 (2007): 277.

APPENDIX

Main script

```
%Hybrid system simulation with two agent case
%using RK method
%Kumaraguru Sivasankaran 04 April 2017
close all;
clear all;
clc;

%bounds on control input
udes=3;
u2des=3.5;

%step size
h=0.005;
t=0:h:10;
p1=zeros(2,length(t));
v1=zeros(2,length(t));
p2=zeros(2,length(t));
v2=zeros(2,length(t));

%initial conditions
p1(:,1)=[0;0];
v1(:,1)=[0;0];
p2(:,1)=[10;0];
v2(:,1)=[0;0];

%define destination point
r1=[10;10;0;0];
r2=[0;10;0;0];
R1=[r1(1);r1(2)];
R2=[r2(1);r2(2)];
%define safety circle radius
d1=2;       %if this is changed, change inside the function as well
%define influence circle radius
d2=1;       %if this is changed, change inside the function as well

%solve for point of intersection tau
A=[p1(:,1)-R1 p2(:,1)-R2];
B=p2(:,1)-p1(:,1);
C=A\B;                              %C=[mu;t]
tau=p1(:,1)+C(1)*(p1(:,1)-R1);

%find secondary stop points
d21=p1(:,1)+(norm(tau-p1(:,1))-d2)*((norm(tau-p1(:,1)))^-1)*(tau-p1(:,1));
d22=p2(:,1)+(norm(tau-p2(:,1))-d2)*((norm(tau-p2(:,1)))^-1)*(tau-p2(:,1));

%functions
F_1 = @(t,w,x,y,z) y;                                   % change the function as you desire
G_1 = @(t,w,x,y,z) z;
F_2 = @(t,w,x,y,z) y;                                   % change the function as you desire
G_2 = @(t,w,x,y,z) z;

%solver RK4 method
for i=1:(length(t)-1)                               % calculation loop
        %calculate relative velocity
        relp=p1(:,i)-p2(:,i);
        relv=v1(:,i)-v2(:,i);
```

```
delv=relp.'*relv/norm(relp);
%calculate the current distance to target
dd1=(((p1(1,i)-tau(1))^2)+(p1(2,i)-tau(2))^2)^0.5;
dd2=(((p2(1,i)-tau(1))^2)+((p2(2,i)-tau(2))^2))^0.5;
%agent1
%RK4 constants
K_1 = F_1(t(i),p1(1,i),p1(2,i),v1(1,i),v1(2,i));
L_1 = G_1(t(i),p1(1,i),p1(2,i),v1(1,i),v1(2,i));
M_1 = H_1(t(i),p1(1,i),p1(2,i),v1(1,i),v1(2,i),dd1,dd2,d21,...
    d22,r1,r2,delv);
N_1 = I_1(t(i),p1(1,i),p1(2,i),v1(1,i),v1(2,i),dd1,dd2,d21,...
    d22,r1,r2,delv);

K_2 = F_1(t(i)+0.5*h,p1(1,i)+0.5*h*K_1,p1(2,i)+0.5*h*L_1,v1(1,i)...
    +0.5*h*M_1,v1(2,i)+0.5*h*N_1);
L_2 = G_1(t(i)+0.5*h,p1(1,i)+0.5*h*K_1,p1(2,i)+0.5*h*L_1,v1(1,i)...
    +0.5*h*M_1,v1(2,i)+0.5*h*N_1);
M_2 = H_1(t(i)+0.5*h,p1(1,i)+0.5*h*K_1,p1(2,i)+0.5*h*L_1,v1(1,i)...
    +0.5*h*M_1,v1(2,i)+0.5*h*N_1,dd1,dd2,d21,d22,r1,r2,delv);
N_2 = I_1(t(i)+0.5*h,p1(1,i)+0.5*h*K_1,p1(2,i)+0.5*h*L_1,v1(1,i)...
    +0.5*h*M_1,v1(2,i)+0.5*h*N_1,dd1,dd2,d21,d22,r1,r2,delv);

K_3 = F_1(t(i)+0.5*h,p1(1,i)+0.5*h*K_2,p1(2,i)+0.5*h*L_2,v1(1,i)+...
    0.5*h*M_2,v1(2,i)+0.5*h*N_2);
L_3 = G_1(t(i)+0.5*h,p1(1,i)+0.5*h*K_2,p1(2,i)+0.5*h*L_2,v1(1,i)+...
    0.5*h*M_2,v1(2,i)+0.5*h*N_2);
M_3 = H_1(t(i)+0.5*h,p1(1,i)+0.5*h*K_2,p1(2,i)+0.5*h*L_2,v1(1,i)+...
    0.5*h*M_2,v1(2,i)+0.5*h*N_2,dd1,dd2,d21,d22,r1,r2,delv);
N_3 = I_1(t(i)+0.5*h,p1(1,i)+0.5*h*K_2,p1(2,i)+0.5*h*L_2,v1(1,i)+...
    0.5*h*M_2,v1(2,i)+0.5*h*N_2,dd1,dd2,d21,d22,r1,r2,delv);

K_4 = F_1(t(i)+h,p1(1,i)+K_3*h,p1(2,i)+L_3*h,v1(1,i)+M_3*h,...
    v1(2,i)+N_3*h); % Corrected
L_4 = G_1(t(i)+h,p1(1,i)+K_3*h,p1(2,i)+L_3*h,v1(1,i)+M_3*h,...
    v1(2,i)+N_3*h);
M_4 = H_1(t(i)+h,p1(1,i)+K_3*h,p1(2,i)+L_3*h,v1(1,i)+M_3*h,...
    v1(2,i)+N_3*h,dd1,dd2,d21,d22,r1,r2,delv); % Corrected
N_4 = I_1(t(i)+h,p1(1,i)+K_3*h,p1(2,i)+L_3*h,v1(1,i)+M_3*h,...
    v1(2,i)+N_3*h,dd1,dd2,d21,d22,r1,r2,delv);

p1(1,i+1) = p1(1,i) + (1/6)*(K_1+2*K_2+2*K_3+K_4)*h;  % main equation
p1(2,i+1) = p1(2,i) + (1/6)*(L_1+2*L_2+2*L_3+L_4)*h;  % main equation
v1(1,i+1) = v1(1,i) + (1/6)*(M_1+2*M_2+2*M_3+M_4)*h;  % main equation
v1(2,i+1) = v1(2,i) + (1/6)*(N_1+2*N_2+2*N_3+N_4)*h;  % main equation

if (v1(1,i+1)-udes)>=1e-4
    v1(1,i+1)=udes;
elseif (v1(1,i+1)+udes)<=1e-4
    v1(1,i+1)=-udes;
else
    v1(1,i+1)=v1(1,i+1);
end

if (v1(2,i+1)-udes)>=1e-4
    v1(2,i+1)=udes;
elseif (v1(2,i+1)+udes)<=1e-4
    v1(2,i+1)=-udes;
else
    v1(2,i+1)=v1(2,i+1);
```

```
end

%agent 2
%RK4 constants
KK_1 = F_2(t(i),p2(1,i),p2(2,i),v2(1,i),v2(2,i));
LL_1 = G_2(t(i),p2(1,i),p2(2,i),v2(1,i),v2(2,i));
MM_1 = H_2(t(i),p2(1,i),p2(2,i),v2(1,i),v2(2,i),dd1,dd2,d21,d22,...
    r1,r2,delv);
NN_1 = I_2(t(i),p2(1,i),p2(2,i),v2(1,i),v2(2,i),dd1,dd2,d21,d22,...
    r1,r2,delv);

KK_2 = F_2(t(i)+0.5*h,p2(1,i)+0.5*h*KK_1,p2(2,i)+0.5*h*LL_1,v2(1,i)...
    +0.5*h*MM_1,v2(2,i)+0.5*h*NN_1);
LL_2 = G_2(t(i)+0.5*h,p2(1,i)+0.5*h*KK_1,p2(2,i)+0.5*h*LL_1,v2(1,i)...
    +0.5*h*MM_1,v2(2,i)+0.5*h*NN_1);
MM_2 = H_2(t(i)+0.5*h,p2(1,i)+0.5*h*KK_1,p2(2,i)+0.5*h*LL_1,v2(1,i)...
    +0.5*h*MM_1,v2(2,i)+0.5*h*NN_1,dd1,dd2,d21,d22,r1,r2,delv);
NN_2 = I_2(t(i)+0.5*h,p2(1,i)+0.5*h*KK_1,p2(2,i)+0.5*h*LL_1,v2(1,i)...
    +0.5*h*MM_1,v2(2,i)+0.5*h*NN_1,dd1,dd2,d21,d22,r1,r2,delv);

KK_3 = F_2(t(i)+0.5*h,p2(1,i)+0.5*h*KK_2,p2(2,i)+0.5*h*LL_2,v2(1,i)...
    +0.5*h*MM_2,v2(2,i)+0.5*h*NN_2);
LL_3 = G_2(t(i)+0.5*h,p2(1,i)+0.5*h*KK_2,p2(2,i)+0.5*h*LL_2,v2(1,i)...
    +0.5*h*MM_2,v2(2,i)+0.5*h*NN_2);
MM_3 = H_2(t(i)+0.5*h,p2(1,i)+0.5*h*KK_2,p2(2,i)+0.5*h*LL_2,v2(1,i)...
    +0.5*h*MM_2,v2(2,i)+0.5*h*NN_2,dd1,dd2,d21,d22,r1,r2,delv);
NN_3 = I_2(t(i)+0.5*h,p2(1,i)+0.5*h*KK_2,p2(2,i)+0.5*h*LL_2,v2(1,i)...
    +0.5*h*MM_2,v2(2,i)+0.5*h*NN_2,dd1,dd2,d21,d22,r1,r2,delv);

KK_4 = F_2(t(i)+h,p2(1,i)+KK_3*h,p2(2,i)+LL_3*h,v2(1,i)+MM_3*h,...
    v2(2,i)+NN_3*h); % Corrected
LL_4 = G_2(t(i)+h,p2(1,i)+KK_3*h,p2(2,i)+LL_3*h,v2(1,i)+MM_3*h,...
    v2(2,i)+NN_3*h);
MM_4 = H_2(t(i)+h,p2(1,i)+KK_3*h,p2(2,i)+LL_3*h,v2(1,i)+MM_3*h,...
    v2(2,i)+NN_3*h,dd1,dd2,d21,d22,r1,r2,delv); % Corrected
NN_4 = I_2(t(i)+h,p2(1,i)+KK_3*h,p2(2,i)+LL_3*h,v2(1,i)+MM_3*h,...
    v2(2,i)+NN_3*h,dd1,dd2,d21,d22,r1,r2,delv);

p2(1,i+1) = p2(1,i) + (1/6)*(KK_1+2*KK_2+2*KK_3+KK_4)*h;  % main equation
p2(2,i+1) = p2(2,i) + (1/6)*(LL_1+2*LL_2+2*LL_3+LL_4)*h;  % main equation
v2(1,i+1) = v2(1,i) + (1/6)*(MM_1+2*MM_2+2*MM_3+MM_4)*h;  % main equation
v2(2,i+1) = v2(2,i) + (1/6)*(NN_1+2*NN_2+2*NN_3+NN_4)*h;  % main equation

if (v2(1,i+1)-u2des)>=1e-4
    v2(1,i+1)=u2des;
elseif (v2(1,i+1)+u2des)<=1e-4
    v2(1,i+1)=-u2des;
else
    v2(1,i+1)=v2(1,i+1);
end

if (v2(2,i+1)-u2des)>=1e-4
    v2(2,i+1)=u2des;
elseif (v2(2,i+1)+u2des)<=1e-4
    v2(2,i+1)=-u2des;
else
    v2(2,i+1)=v2(2,i+1);
end
```

```
        rel(i)=delv;
end

v1(3,:) = ((v1(1,:).^2) + (v1(2,:).^2)).^0.5;
v2(3,:) = ((v2(1,:).^2) + (v2(2,:).^2)).^0.5;
p1(3,:) = (((p1(1,:)-p1(1,1)).^2) + ((p1(2,:)-p1(2,1)).^2)).^0.5;
p2(3,:) = (((p2(1,:)-p2(1,1)).^2) + ((p2(2,:)-p2(2,1)).^2)).^0.5;

%velocity plot
figure
grid on
subplot(4,1,1)
plot(t,v1(3,:));
xlabel('time');
ylabel('velocity of agent1');
title('velocity vs time')

%velocity plot
grid on
subplot(4,1,2)
plot(t,v2(3,:));
xlabel('time');
ylabel('velocity of agent2');
title('velocity vs time')
grid on

%position plot
grid on
subplot(4,1,3)
plot(t,p1(3,:));
xlabel('time');
ylabel('position of agent1');
title('Distance vs time')
grid on
subplot(4,1,4)
plot(t,p2(3,:));
xlabel('time');
ylabel('position of agent2');
title('Distance vs time')

%error plot
figure
grid on
error=p1-p2;
error(3,:)=(error(1,:).^2+error(2,:).^2).^0.5;
plot(t,error(3,:))
xlabel('time')
ylabel('Distance between agents')
title('Collision check plot')
grid on
%comment below
figure

%plot the circles
grid on
centers=tau';
radii=d1;
viscircles(centers,radii);
hold on;
```

```
centers=tau';
radii=d2;
viscircles(centers,radii);
hold on;
%plot the points
scatter(tau(1),tau(2),'filled')
hold on
scatter(d21(1),d21(2),'filled')
hold on
scatter(d22(1),d22(2),'filled')
hold on
scatter(r2(1),r2(2),'filled')
hold on
scatter(r1(1),r1(2),'filled')
hold on
scatter(p1(1,1),p1(2,1),'filled')
hold on
scatter(p2(1,1),p2(2,1),'filled')
xlim([-5 15])
ylim([-5 15])
xlabel('position in x coordinates')
ylabel('position in y coordinates')
title('position plot')
axis square
axis([-5,15,-5,15])
High=animatedline('Color','r');
Low=animatedline('Color','b');
for i=1:length(t)
    addpoints(High,p1(1,i),p1(2,i));
    drawnow
    addpoints(Low,p2(1,i),p2(2,i));
    drawnow
    pause(0.00001)
    grid on
end
figure
rel(i)=rel(i-1);
plot(t,rel)
hold on
yyy=zeros(length(t));
plot(t,yyy);
hold on
plot(t,error(3,:))
hold on
plot(t,p1(3,:))
hold on
plot(t,p2(3,:))
hold on
plot(t,v1(3,:))
hold on
plot(t,v2(3,:))
hold on
distance2=linspace(2,2,length(t));
plot(t,distance2)
distance1=linspace(1,1,length(t));
hold on
plot(t,distance1)
```

Controller 1:

```
%controller for agent 1
```

```
function l=I_1(t,w,x,y,z,dd1,dd2,d21,d22,r1,r2,delv)
%predesigned control gains
k1=1;
k2=1.732;
d1=4;
d2=2;
udes=3;
if dd1^2 > d1^2
    l=-k1*(x-r1(2))-k2*(z-r1(4));
elseif dd1^2 < d2^2
    l=-k1*(x-r1(2))-k2*(z-r1(4));
else
    if ((dd2^2 <= d2^2) && (delv < 1e-4))
        l=-k1*(x-d21(2))-k2*(z-r1(4));
    else
        l=-k1*(x-r1(2))-k2*(z-r1(4));
    end
end

if (l-udes)>=1e-4
    l=udes;
elseif (l+udes)<=1e-4
    l=-udes;
else
    l=l;
end
end
```

```
%controller for agent 1
function h=H_1(t,w,x,y,z,dd1,dd2,d21,d22,r1,r2,delv)
%predesigned control gains
k1=1;
k2=1.732;
d1=4;
d2=2;
udes=3;
if dd1^2 > d1^2
    h=-k1*(w-r1(1))-k2*(y-r1(3));
elseif dd1^2 < d2^2
    h=-k1*(w-r1(1))-k2*(y-r1(3));
else
    if ((dd2^2 <= d2^2) && (delv < 1e-4))
        h=-k1*(w-d21(1))-k2*(y-r1(3));
    else
        h=-k1*(w-r1(1))-k2*(y-r1(3));
    end
end

if (h-udes)>=1e-4
    h=udes;
elseif (h+udes)<=1e-4
    h=-udes;
else
    h=h;
end
end
```

Controller 2:

```
%controller for agent 2
function n=I_2(t,w,x,y,z,dd1,dd2,d21,d22,r1,r2,delv)
```

```
%predesigned control gains
k1=1;
k2=1.732;
d1=4;
d2=2;
u2des=3.5;
if dd2^2 > d1^2
    n=-k1*(x-r2(2))-k2*(z-r2(4));
elseif dd2^2 < d2^2
    n=-k1*(x-r2(2))-k2*(z-r2(4));
else
    if ((dd1^2 <= d2^2)&&(delv < 1e-4))
        n=-k1*(x-d22(2))-k2*(z-r2(4));
    else
        n=-k1*(x-r2(2))-k2*(z-r2(4));
    end
end

if (n-u2des)>=1e-4
    n=u2des;
elseif (n+u2des)<=1e-4
    n=-u2des;
else
    n=n;
end
end

%controller for agent 2
function m=H_2(t,w,x,y,z,dd1,dd2,d21,d22,r1,r2,delv)
%predesigned control gains
k1=1;
k2=1.732;
d1=4;
d2=2;
u2des=3.5;
if dd2^2 > d1^2
    m=-k1*(w-r2(1))-k2*(y-r2(3));
elseif dd2^2 < d2^2
    m=-k1*(w-r2(1))-k2*(y-r2(3));
else
    if ((dd1^2 <= d2^2)&&(delv < 1e-4))
        m=-k1*(w-d22(1))-k2*(y-r2(3));
    else
        m=-k1*(w-r2(1))-k2*(y-r2(3));
    end
end

if (m-u2des)>=1e-4
    m=u2des;
elseif (m+u2des)<=1e-4
    m=-u2des;
else
    m=m;
end
end
```