ENEE322 Final Project

Kyle Montemayor, Ian Wilmoth, and William Zhang

***Abstract*—Controlling fleets of autonomous vehicles presents a challenge in that overall all of the vehicles represent a very large system, of which a control law is hard to develop. However, given our increasing propensity towards autonomy, being able to control and model a fleet of such systems is crucial to our society.**

I. Introduction

Vehicles with a differential drive are very common in a small form factor. Due to their ability to have a very low turning radius, they are very maneuverable. Unfortunately they are not well suited to larger tasks as they require additional drive motors to be most efficient.

For this paper, a procedure for controlling a fleet of the autonomous differential drive vehicles(ADDV) such that they can locate each other, form a single file, and follow a path in the formation. This will be accomplished in 4 parts, simulating a ADDV, controlling an ADDV to a desired end point, controlling an ADDV along a desired path, having two ADDVs follow each other, and finally, having a fleet of ADDVs follow each other.

II. Autonomous Differential Drive Vehicles

The vehicle's that are studied in this paper have a drive mechanism as shown below.

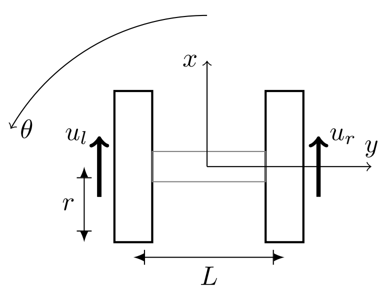

Fig. 1. Diagram of ADDV drive mechanism

Where the controls, given by u_r, u_l, are for the left and right motors, r is the radius of the wheels, L is the distance between the wheels, (x, y) are Cartesian coordinates of the center of the ADDV, and θ is the angle between r and x. All together, the state vector $(x(t), y(t), \theta(t))$ is the pose of the ADDV at time t.

A. Control Scheme

With the given definitions we can define the equations of motion of the as follows

$$\dot{x} = \frac{r}{2}(u_r + u_l)cos(\theta) \tag{1}$$

$$\dot{y} = \frac{r}{2}(u_r + u_l)sin(\theta) \tag{2}$$

$$\dot{\theta} = \frac{r}{L}(u_r - u_l) \tag{3}$$

Where $\theta = 0$ is located in the $+x$ direction. Clearly, the equations for $(\dot{x}, \dot{y})$ are non-linear, but if we define

$$\dot{r} = \sqrt{\dot{x}^2 + \dot{y}^2} = \frac{r}{2}(u_r + u_l) \tag{4}$$

and use a polar coordinate system originating from the origin of the ADDV then we have a completely linear system of equations given by 3, 4.

This can be a useful coordinate system as we can define all desired end points as polar coordinates originating from ADDV. In order to additionally simplify our controls, we will define our translational and rotational movement as follows

$$\dot{r} = \frac{r}{2}(u_r + u_l), u_r = u_l \tag{5}$$

$$\dot{\theta} = \frac{r}{L}(u_r - u_l), u_r = -u_l \tag{6}$$

We will also only employ one form of movement at a time, either translational (5), or rotational (6).

It should be noted that a more complicated control scheme could be developed, and have the ADDV's move in more graceful lines. Having $u_l > u_r > 0$ would cause the ADDV to move in a clockwise circle. By adjusting the difference between the inputs you could have the ADDV end up in most poses. Noticable poses that you cannot reach by mearly traveling in a circle would be from $(0, 0, \pi/2)$ to $(0, 1, -\pi/2)$ this could potentially be solved by having inputs of oposing sign. However these solutions would not give the shortest distance traveled, or potentially the shortest amount of time.

III. Procedure

A. Modelling/Simulation

We used the MATLAB simulation provided to us with the project as a basis for the simulation. It works by plotting the location of each ADDV at each time step. We use the MATLAB ODE solver to solve for the points given our computed inputs.

B. Control of One ADDV

If we want to move our ADDV from (x_0, y_0, θ_0) to some $(\bar{x}, \bar{y}, \bar{\theta})$ then a simple, if inelegant solution is to break our movement into 3 steps

1) Rotate until θ is pointed towards our desired location
2) Translate until the ADDV has arrived at the destination
3) Rotate again to match θ with $\bar{\theta}$

The matlab code for this fucntion is given in the appendix.

C. Control of One ADDV Along Path

We can imagine any given path $(x(t), y(t), \theta(t))$ as a series of discrete points. By breaking up the path, we can repeat the steps of the second part of our problem to have and ADDV follow the path.

D. Pair of ADDV

This should be done in 2 steps. The first should be to have the two ADDV's assemple into their desired formation. Then, we can feed the control vectors to the lead ADDV and a copy of the control vectors padded with 1 zero to the second ADDV.

E. Fleet of ADDV

We would go about this by first organizing the ADDV's into the formation. Finding the line (formation) such that the distance traveled by the ADDV's is a rather dificult problem, but a much easier solution is to have all of the ADDV's line up behind the lead ADDV, in a order predertermined by either our simulation or by the software running on the ADDV's before assembling. From their, we can treat the problem as a set of many pairs of the same problem solved by 4. We would have our control vectors be delayed by the number of the ADDV, using a zero based counting system from the lead ADDV.

IV. Discussion

Appendix

```
% returns controls and positions to move from the
% current pose (x_0, y_0, o_0) to the desired end
% currently DOES NOT account for max speed/errors
% or RETURN x, y, o VALUES (im lazy)
function [u_l, u_r, x, y, o] = move_addv(x_0, y_0
    r = 1; % radius of wheels
    L = 1; % radius of car

    % first rotate to face x_f, y_f
    theta = o_f - atan(y_o/x_o); % Need to have c
    u_l(1) = L*2/r * theta;
    u_r(1) = - u_l(1);

    % now move to x_f, y_f
    d = sqrt((x_f-x_0)^2 + (y_f-y_0)^2);
    u_l(2) = 4/r * d;
    u_r(2) = u_l(2);

    % now face to o_f

    theta = o_f - theta;
    u_l(3) = L*2/r * theta;
    u_r(3) = - u_l(3);
end
```