Centralised Controller for Rigid Formations in AGV fleets

Thomas Udo, Jordi Espina Font, Sara van Eersel, Cas Leeflang, Mart Coppelmans

July 4, 2023

Contents

1 Introduction

In this document the developed controller to move an arbitrary number of differential drive robots in a formation is described; thus creating a Multi-Robot System (MRS). The goal is to move the MRS as a Virtual Structure (VS), which is a rigid formation that makes it look as if there was a rigid mechanical linkage between all members; they do not change their positions relative to each other throughout their movements.

The controller (Figure 1) receives a VS path as an input; receives the poses of the robots as feedback; and outputs the required angular and rotational velocities for each robot for the VS to follow its desired path.

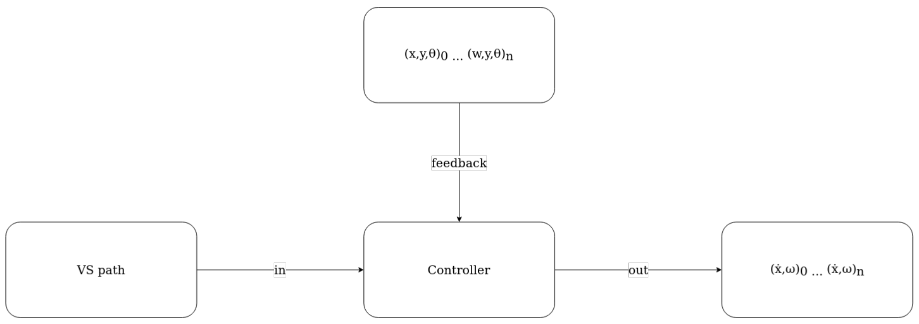

Figure 1: Controller diagram

2 Terms and abbreviations

1. AGV - Autonomous Guided Vehicle
2. VS - Virtual Structure
3. Bot - Robot
4. MRS - Multi-Robot System

3 Notation

The x, y and θ of a pose relative to a frame are expressed as follows.

$$s_{target}^{reference}$$

If either the superscript or the subscript are omitted, it can be assumed it refers to the world frame $\{W\}$.

$$C_g = C_g^W, D^t = D_W^t$$

4 Robot Routes

Given the desired path (as a sequence of poses relative to the world frame) for the VS, the route for each robot needs to be computed. Note that there is only one solution that maintains rigidity.

The route of a bot is the sequence of all the coordinates it must follow. It is derived from the VS path. Note that bot orientation is irrelevant to the pose of the VS.

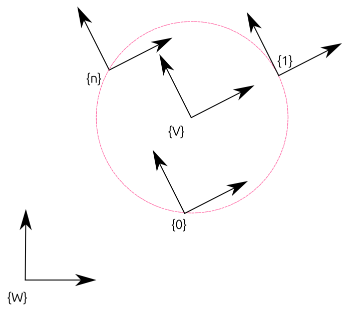

Figure 2: Reference frames

As Figure 2 illustrates, the frames of the robots are described by an index starting from 0 to n. The frame of the VS is named V.

4.1 Robot pose transformation

The coordinates of the robots relative to the VS can be calculated from the amount of robots in the formation (N) and the radius (q) of the formation, with the index of the robot (n)

$$x_n = \cos \frac{\tau n}{N} \cdot q$$

$$y_n = \sin \frac{\tau n}{N} \cdot q$$

$$C_n^V = \begin{pmatrix} x_n \\ y_n \end{pmatrix}$$

4.2 Path transformations

The transformation matrix from one of the VS poses in the path to the world frame $\{W\}$ looks as follows.

$$T_{V,i}^W = \begin{bmatrix} \cos\theta_V & -\sin\theta_V & x_V \\ \sin\theta_V & \cos\theta_V & y_V \\ 0 & 0 & 1 \end{bmatrix}_i$$

Finally, to calculate each robot way point.

$$r_{n,i} = T_{V,i}^W \cdot \begin{pmatrix} C_n^V \\ 1 \end{pmatrix}$$

All of which can be encapsulated in one matrix containing all robot routes.

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,i} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,i} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,1} & r_{n,2} & \cdots & r_{n,i} \end{bmatrix}$$

5 Synchronized pure pursuit

The robots require a velocity command (linear $\dot{x}$ and angular velocity ω) as an input. In this section, the method to calculate the required velocities to reach the goal is described. It consist of a modified version of the pure pursuit algorithm. [1]

In this algorithm, only the immediate goal is relevant to the calculation. The goal (g) coordinates are expressed as follows.

$$C_g = \begin{pmatrix} x \\ y \end{pmatrix}$$

Synthesised, the algorithm works as follows.

1. *For each robot:*
   (a) *The goal is computed*
   (b) *The orthodromic distance to the goal is calculated*
2. *The velocities are calculated such that all the trajectories possess the same Δt to completion*
3. *Repeat if goal is not yet reached*

5.1 Carrot goal

5.1.1 Orthogonal projection to route

Instead of calculating the carrot goal from the coordinates of the robots, it is done from their orthogonal projections to the route (C'), which in this case is approximated to the closest way point. The scope of the calculations is limited to the last point and the next few.

$$\Delta C_{n,i} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}_{n,i} = C_n - R_{n,i}$$

$$D_{n,i} = \sqrt{\Delta y_{n,i} + \Delta x_{n,i}}$$

$$C_n' = \min_{i \in I} D_n$$

5.1.2 Look Ahead

The carrot goal is calculated from point C'_n, the look ahead index (L) is given as a parameter. The carrot goal is as follows:

$$C'_{i+L,n}$$

5.1.3 Goal transformation

The coordinates of the goal need to be relative to the robot frame.

Two different transformations are required, the first one for translating the reference frame, and the later one for rotating it.

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}^n = -\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}_n$$

$$T_W^{n'} = \begin{bmatrix} 1 & 0 & x^n \\ 0 & 1 & y^n \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{n'}^n = \begin{bmatrix} \cos\theta_n & -\sin\theta_n & 0 \\ \sin\theta_n & \cos\theta_n & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_W^n = T_{n'}^n \cdot T_W^{n'}$$

Finally, the transformation can be performed.

$$C_g^n = T_W^n \cdot \begin{pmatrix} C_g \\ 1 \end{pmatrix}$$

5.2 Orthodromic Distance

In order to synchronize the arrival of all robots to their goal $(C_g)_n$, the length of their goal trajectory Δs_n is required [1] that is precisely the orthodromic distance (Figure 3).

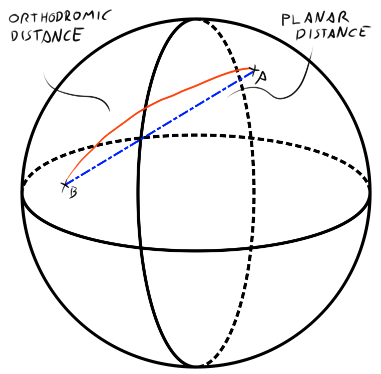

Figure 3: The orthodromic distance is the shortest distance between two points in a spherical surface

The displacement from the goal.

$$l = \sqrt{(x_g^n)^2 + (y_g^n)^2}$$

Get the radius of the path.

$$q = \frac{l^2}{2y_g^n}$$

Calculate the angle between the points relative to the centre of the arc.

$$\sin\frac{\alpha}{2} = \frac{l}{2q}$$

$$\alpha = \frac{x}{|x|} 2 \arcsin\frac{l}{2q}$$

With r and θ the perimeter can be computed.

$$\Delta s = q\alpha$$

Because the path is followed tangentially. The angle of the circular path segment is equal to the change of angle in the robot's pose $\Delta\theta = \alpha$ (Figure 4).

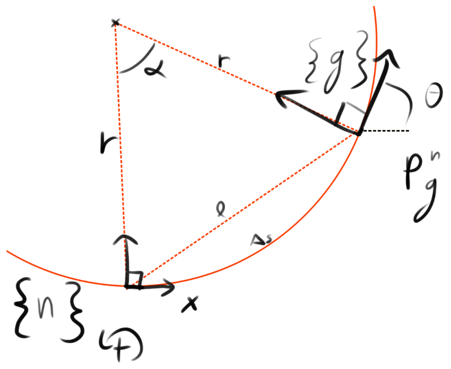

Figure 4: Arc trajectory

5.3 Velocity Calculation

The robots move synchronously when they all share the same Δt from their goals. The robot with the largest Δs (*maxpath*) determines the velocity of the other robots, since it is desired that the robots always reach the maximum allowable speed.

For calculating *maxpath*, $q = (q_{arc} + q_{wheelbase}\frac{q_{arc}}{|q_{arc}|})$, this means that for *maxpath* is $\Delta s'$ and $\Delta\theta'$

$$\Delta t = \left|\frac{maxpath}{maxwheelvelocity}\right| = \left|\frac{\Delta s'}{maxwheelvelocity}\right|$$

The robot with the largest $\Delta s'$ receives velocity *max speed*, the other robots receive velocities directly proportional to their local route length.

$$\dot{x}_n = \frac{\Delta s_n}{\Delta t}, \omega_n = \frac{\Delta \theta_n}{\Delta t}$$

References

[1] R. Craig Coulter. “Implementation of the Pure Pursuit Path Tracking Algorithm”. In: 1992.