Single Robot Triangle Formation

Yuhan Zhao

April 23

1 Problem Setting

Assume we have a robot A, a human B, and a fixed object C shown in the following figure. The objective is to form an equilateral triangle $\triangle ABC$ by controlling the robot to the proper position after observing the human's position.

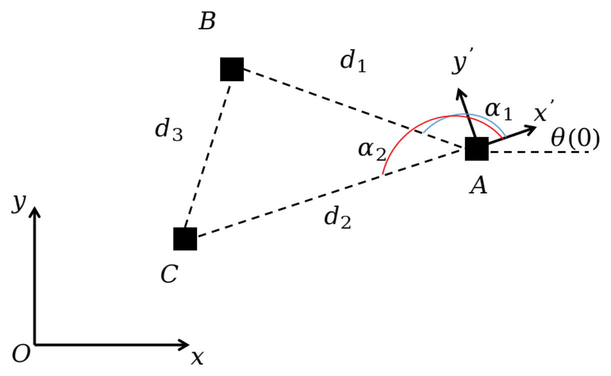


Figure 1: Sketch of triangle formation.

From the robot Lidar we can obtain the angle α_1, α_2 and the distance $d_1 = |AB|$, $d_2 = |AC|$. Using cosine law, we can compute the distance

$$d_3 = |BC| = \sqrt{d_1^2 + d_2^2 - 2d_1 d_2 \cos(\alpha_2 - \alpha_1)}. \tag{1}$$

Note: B and C are fixed when the robot is moving. Although d_1, d_2 and α_1, α_2 are changing during the robot's movement, d_3 remains constant. Therefore, we can evaluate d_3 at the beginning of the movement. We denote $d_3 = \bar{d}$.

We aim to design a controller to drive the robot to the desired position A^*. We use the convention that $\alpha_1, \alpha_2 \in [0, 2\pi)$. Thus we have $d_1^* = d_2^* = \bar{d}$ and $\alpha_1^* = -\frac{\pi}{6}$ and $\alpha_2^* = \frac{\pi}{6}$.

We establish two coordinate frame O-xy and A-$x'y'$. We can use the *unicycle model* to model the robot, which is

$$\dot{\theta} = w, \quad \dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \tag{2}$$

where v and w are control variables. Note that the observable states are d_1, d_2, α_1, α_2, which are all described in the local frame A-xy. To make the control easier, we choose d_1, d_2, θ as the system state. Since we have

$$\begin{aligned} d_1 &= \|p_B - p_A\|_2 = \sqrt{(P_{Bx} - x)^2 + (P_{By} - y)^2}, \\ d_2 &= \|p_C - p_A\|_2 = \sqrt{(P_{Cx} - x)^2 + (P_{Cy} - y)^2}, \end{aligned}$$

we have

$$\begin{aligned} \dot{d}_1 &= \frac{1}{d_1}\left((P_{Bx} - x)(-\dot{x}) + (P_{Cx} - x)(-\dot{y})\right) \\ &= -\frac{1}{d_1}\left(d_1 \cos(\theta + \alpha_1) v \cos(\theta) + d_1 \sin(\theta + \alpha_1) v \sin(\theta)\right) \\ &= -\cos(\theta + \alpha_1) v \cos(\theta) - \sin(\theta + \alpha_1) v \sin(\theta) \\ &= -v \cos(\alpha_1). \end{aligned} \tag{3}$$

Likewise,

$$\dot{d}_2 = -v \cos(\alpha_2). \tag{4}$$

To estimate θ from α_1 and α_2, we need to know to additional quantities: $\theta(0)$, $\alpha_1(0)$ and $\alpha_2(0)$. The first one can be always assign to 0 by aligning O-xy and A-xy frame. The last two can be obtained from the beginning when we start the robot. Then we have

$$\theta = \alpha_1(0) - \alpha_1 + \theta(0) = \alpha_2(0) - \alpha_2 + \theta(0). \tag{5}$$

Clearly,

$$\dot{\theta} = -\dot{\alpha}_1 = -\dot{\alpha}_2. \tag{6}$$

From above analysis, we see that it is in fact more convenient to choose d_1, d_2, α_1, α_2 as state variables, we finally reach the following dynamical system:

$$\begin{aligned} \dot{d}_1 &= -v \cos(\alpha_1), \\ \dot{d}_2 &= -v \cos(\alpha_2), \\ \dot{\alpha}_1 &= -w, \\ \dot{\alpha}_2 &= -w. \end{aligned} \tag{7}$$

Let $\mathbf{x} = [d_1 \ d_2 \ \alpha_1 \ \alpha_2]^T$ and $\mathbf{u} = [v \ w]$, we can write (7) as the compact form

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}). \tag{8}$$

Note all states are observable, we do not need the observable function.

To define the objective, since we want to drive the robot to A^*, we use quadratic penalty

$$\begin{aligned} J(\mathbf{x}, \mathbf{u}) &= \left\|d_1 - \bar{d}\right\|_2^2 + \left\|d_2 - \bar{d}\right\|_2^2 + \left\|\alpha_1 - \frac{11\pi}{6}\right\|_2^2 + \left\|\alpha_2 - \frac{\pi}{6}\right\|_2^2 \\ &= \mathbf{x}^T \mathbf{x} + h^T \mathbf{x} + c, \end{aligned} \tag{9}$$

where $f = \begin{bmatrix} -2\bar{d} & -2\bar{d} & -\frac{22\pi}{6} & -\frac{\pi}{3} \end{bmatrix}^T$ and $c = 2\bar{d}^2 + \frac{61\pi^2}{18}$.

We do not consider other constraints such as collision avoidance constraints for now.

2 Controller Design

With the cost (9) and the dynamics (8), we can define the following optimal control problem to drive the robot to A^*:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \mathbf{x}_f^T \mathbf{x}_f + h^T \mathbf{x}_f + c + \int_0^T \mathbf{u}(t)^T \mathbf{u}(t) dt \\ \text{s.t.} \quad & \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \quad \mathbf{x}(0) \text{ given.} \end{aligned} \tag{10}$$

The control cost is added to achieve energy efficient control.

Note that (10) can be solved via maximum principle or dynamic programming. However, the nonlinear dynamics makes it harder to do this. Since there is no additional constraints, we consider the following simplified gradient based controller.

The gradient based controller does not consider the running cost $\int_0^T \mathbf{u}(t)^T \mathbf{u}(t) dt$. It only considers the terminal cost J. Note that

$$\begin{aligned} \dot{J} &= 2(d_1 - \bar{d})\dot{d}_1 + 2(d_2 - \bar{d})\dot{d}_2 + 2(\alpha_1 - \frac{11\pi}{6})\dot{\alpha}_1 + 2(\alpha_2 - \frac{\pi}{6})\dot{\alpha}_2 \\ &= -2(d_1 - \bar{d})v\cos(\alpha_1) - 2(d_2 - \bar{d})v\cos(\alpha_2) - 2(\alpha_1 - \frac{11\pi}{6})w - 2(\alpha_2 - \frac{\pi}{6})w. \end{aligned} \tag{11}$$

We want to choose v and w such that $\dot{J} \leq 0$. This means that the control decreases the cost J according to its time derivative so that it will converges to some local minimum of J. The following v and w can be a candidate:

$$v_c = \frac{1}{2}\left((d_1 - \bar{d})\cos(\alpha_1) + (d_2 - \bar{d})\cos(\alpha_2)\right), \quad w_c = \frac{1}{2}(\alpha_1 + \alpha_2). \tag{12}$$

Taking (12) into (11), we obtain

$$\dot{J} = -\left\|(d_1 - \bar{d})\cos(\alpha_1) + (d_2 - \bar{d})\cos(\alpha_2)\right\|_2^2 - \|\alpha_1 + \alpha_2\|_2^2 \leq 0. \tag{13}$$

Indeed, $\dot{J} = 0$ if and only if $(d_1 - \bar{d})\cos(\alpha_1) + (d_2 - \bar{d})\cos(\alpha_2) = 0$ and $\alpha_1 + \alpha_2 = 2\pi$.

Note: Since the robot dynamics is nonlinear, the gradient controller (12) is not guaranteed to drive the robot to the desired position A^*. Instead, the controller will drive to the local minimum of J, which is one of the element in $S := \{(d_1, d_2, \alpha_1, \alpha_2) \mid (d_1 - \bar{d})\cos(\alpha_1) + (d_2 - \bar{d})\cos(\alpha_2) = 0, \alpha_1 + \alpha_2 = 2\pi\}$. Whether the converged point is close to A^* depends on the initial condition, which will be discussed in the next section.

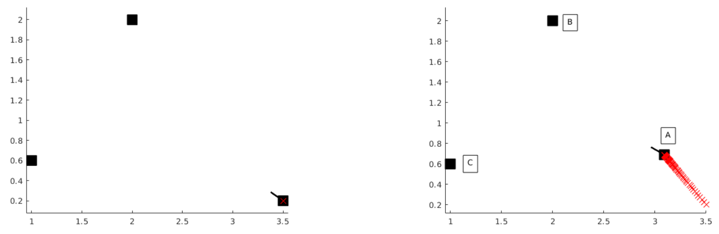
We define the following normalization function

$$\mathrm{vers}(x) = \begin{cases} x & \|x\| \le x_{\max}, \\ \frac{x}{\|x\|} x_{\max} & \|x\| > x_{\max}. \end{cases} \tag{14}$$

In practice, we use $\mathrm{vers}(v_c)$ and $\mathrm{vers}(w_c)$ instead of v_c and w_c in (12) to avoid too large controls.

3 MATLAB Simulation

We pick the initial points $A(3.5, 0.2), B(2, 2), C(1, 0.6)$. We confine $v_{\max} = w_{\max} = 0.5$. The initial pose $\theta_0 = 145°$. The initial and terminal plots are shown as follows.



(a) Initial configuration. (b) Terminal configuration.

Figure 2: Simulation trajectory for the triangle formation.

3.1 About convergence to local minimum

As we discussed before, the gradient based controller will drive the robot to one of the local minimum of J, not necessarily A^*. Therefore, in practice, we need to design the cost function J and pick up the initial condition carefully so that the converged point is close to A^*.

The initial condition can be picked near the optimal point A^*. If the initial condition is far away from A^*, we may never converge to a local minimum which is close to A^*. This is the reason why we pick up the value for the simulation at the beginning of the section.

The cost function is already defined in (9). We want to modify it a little bit so that the vanishing point of its derivative $\dot{J}$ will be close to A^*. For any element $(d_1, d_2, \alpha_1, \alpha_2) \in S$, we note that

$$\alpha_1 + \alpha_2 = 2\pi \quad \Rightarrow \quad \cos(\alpha_1) = \cos(\alpha_2),$$

and

$$(d_1 - \bar{d})\cos(\alpha_1) + (d_2 - \bar{d})\cos(\alpha_2) = 0 \quad \Rightarrow \quad \frac{d_1 - \bar{d}}{d_2 - \bar{d}} = -1.$$

Therefore, gradient based controller may drive to the point such that $d_1 < \bar{d}$, $d_2 > \bar{d}$ and $(\bar{d} - d_1) = (d_2 - \bar{d})$. Clearly, A^* is one of those points. We can introduce a parameter $\gamma > 0$ to make the local minimum close to A^* by redefining the cost function in (9) as

$$J(\mathbf{x}, \mathbf{u}) = \left\|d_1 - \bar{d}\right\|_2^2 + \gamma \left\|d_2 - \bar{d}\right\|_2^2 + \left\|\alpha_1 - \frac{11\pi}{6}\right\|_2^2 + \left\|\alpha_2 - \frac{\pi}{6}\right\|_2^2. \tag{15}$$

In this way, we still have $\alpha_1 + \alpha_2 = 2\pi$ at local optima, but we also have

$$d_1 - \bar{d} = \gamma(\bar{d} - d_2).$$

The simulation shows that $\gamma = 0.1$ can help the gradient controller drive the robot to local minimum which is closer to A^*.

The code can be found at `https://yuhan-zhao.github.io/larx_resource`.