AA274A: Principles of Robot Autonomy I
Course Notes

Oct 3, 2019

4 Trajectory Tracking and Closed-loop Control

This lecture delves deeply into topics in trajectory tracking and closed-loop control. First, we build upon our previous discussion on differential flatness to explore how we can obtain exact trajectories for differentially flat systems. Next, we quickly survey ways to derive a closed-loop control law on nonlinear systems. We will end our discussion by deriving a closed-loop control law to solve a pose stabilization problem using Lyapunov stability analysis.

4.1 Bound constraints and time scaling

We introduced the concept of differential flatness in Lecture 3. The implications of flatness is that the trajectory generation problem can be reduced to simple algebra, in theory, and computationally attractive algorithms in practice, since the feasible trajectories of the system are completely characterized by the state variables. [Mur] By the nature of differentially flat systems, constraints can be transformed into the flat output space and typically become limits on the curvature, or higher order derivative properties of the trajectory.

While there can be several different types of constraints imposed on the system, one of the important classes of constraints are *bound constraints*. For example, our robot could have an upper bound for linear and angular velocity, i.e:

$$|V(t)| \leq V_{max}, |\omega(t)| \leq \omega_{max}$$

One way to ensure that a computed trajectory meets this constraint is to use *time rescaling*. The key idea is to decompose the planned trajectory into a geometric *path*, that defines the sequence of configurations taken by the robot, and a *time-scaling law*, which specifies the times these configurations are reached. Mathematically, we break down the trajectory $\boldsymbol{x}(t)$ into a path $\tilde{\boldsymbol{x}}(s)$ and a timing law $s(t)$, where s is a parameter that varies from $s(t_0) = s_0$ to $s(t_f) = s_f$ in a monotonic fashion, implying $\dot{s}(t) > 0$. A common choice for s is the arc length from the start of the path to the point along the path, meaning s varies from $s_0 = 0$ to $s_f = L$ where L is the length of the path.

This decomposition implies that

$$\dot{\boldsymbol{x}}(t) = \frac{d\boldsymbol{x}(t)}{dt} = \left.\frac{d\tilde{\boldsymbol{x}}(s)}{ds}\right|_{s=s(t)} \frac{d}{dt}s(t). \tag{1}$$

Thus, *after* computing the original trajectory, we can alter the time scaling law $s(t)$ while preserving the path $\tilde{\boldsymbol{x}}(s)$. Changing $s(t)$ will alter the time derivatives of the flat output state $\dot{\boldsymbol{x}}(t)$, and thus change the values of the controls $\boldsymbol{u}(t)$, which are computed from the flat outputs and their derivatives.

More concretely, let's consider the unicycle model, where the controls are given by

$$V(t) = \frac{d||\boldsymbol{x}||}{dt} = \left.\frac{d||\tilde{\boldsymbol{x}}||}{ds}\right|_{s=s(t)} \frac{d}{dt}s(t) \tag{2}$$

$$\omega(t) = \frac{d\theta(\boldsymbol{x}(t))}{dt} = \left.\frac{d\theta(\tilde{x}(s))}{ds}\right|_{s=s(t)} \frac{d}{dt}s(t) \tag{3}$$

Now, we are free to alter the controls $V(t)$ and $\omega(t)$ by simply choosing a new mapping $s_*(t)$, as long as the mapping is monotonically increasing from s_0 to s_f. For example, treating s as the arc-length, a simple approach would be to choose $s_*(t) = \alpha t L/T$, where L, T are the length and original timelength of the original trajectory. By choosing $\alpha < 1$, we can globally slow down the rate at which we traverse the geometric path in order to satisfy the bound constraints.

4.2 Trajectory tracking

In Chapter 3, we introduced a two-step design where closed-loop solution to a trajectory tracking compensates the lack of accuracy in open-loop control. Two-step design can be expressed as:

$$\mathbf{u}^*(t) = \mathbf{u}_d(t) + \pi(\mathbf{x}(t) - \mathbf{x}_d(t), t) \tag{4}$$

where the reference trajectory, $\boldsymbol{x}_d(t)$, and control history, $\boldsymbol{u}_d(t)$, are derived as an open-loop solution of a differentially flat system. The tracking policy, π, is a closed-loop solution to a tracking problem that keeps our trajectory as close as possible to the nominal (or 'reference' trajectory. In this section, we will discuss closed-loop solutions in the context of trajectory tracking in greater detail.

4.2.1 Trajectory tracking strategies

There exists a certain number of choices of techniques for trajectory tracking with nonlinear system:

- *Geometric strategies*:
  Pure pursuit is a path tracking algorithm. It computes the angular velocity command

that moves the robot from its current position to reach some look-ahead point in front of the robot. The linear velocity is assumed constant, hence you can change the linear velocity of the robot at any point. The algorithm then moves the look-ahead point on the path based on the current position of the robot until the last point of the path. You can think of this as the robot constantly chasing a point in front of it [201]. This method is effective, but it is difficult to provide theoretical guarantees.

- *Linearization techniques*:
  At a high level, *inexact linearization* models the first few terms of system dynamics in Taylor series expansion to find a linear control law. *Exact linearization*, on the other hand, cancels the nonlinearities in the system dynamics, and develop a tracking control law from the resulting linear system. ENGR209A discusses linearization techniques on nonlinear systems in greater depth.

- *Optimization based techniques*:
  Tracking problems can be cast as an optimal control problem that can be solved via indirect and direct methods. These techniques are thoroughly studied in AA203.

4.2.2 Trajectory tracking for differentially flat systems

Levine [RPK13, Lev09] shows that every flat system can be linearized to a dynamic feedback form:

$$\boldsymbol{z}^{(q+1)} = \boldsymbol{w}, \tag{5}$$

where $\boldsymbol{z}$ is new configuration variables, q is the degree of new output space (i.e., how many derivatives of the flat output that were needed to describe system dynamics), and $\boldsymbol{w}$ is our virtual control input.

Therefore, we can design a tracking controller for the linearized system by using linear control techniques. In particular, for a given reference flat output $\boldsymbol{z}_d$, define the component-wise error:

$$e_i := z_i - z_{i,d} \tag{6}$$

which, based on (5), implies:

$$e_i^{(q+1)} := w_i - w_{i,d}. \tag{7}$$

A control law which is proportional to the error and its derivatives,

$$w_i = w_{i,d} - \sum_{j=0}^{q} k_{i,j} e_i^{(j)}, \tag{8}$$

is a common choice for guaranteed convergence to zero of the tracking error. The gains $k_{i,j}$ are tuned to ensure stability and tune performance.

For example if the number of degrees is $q = 2$, then the virtual control becomes the second derivative of the flat outputs:

$$\boldsymbol{w} := \ddot{\boldsymbol{z}} \tag{9}$$

and the tracking controller can be expressed as

$$\boldsymbol{w} = \ddot{\boldsymbol{z}} + \boldsymbol{k}_p(\boldsymbol{z}_d - \boldsymbol{z}) + \boldsymbol{k}_d(\dot{\boldsymbol{z}}_d - \dot{\boldsymbol{z}}). \tag{10}$$

4.3 Closed-loop Control

In the case of closed-loop control, the goal is to find

$$\boldsymbol{u}^*(t) = \pi(\boldsymbol{x}(t), t). \tag{11}$$

For stability analysis and stabilization problems of nonlinear systems, *Lyapunov functions* are widely used. The direct Lyapunov method is a generalization of the energy concepts associated with a mechanical system: the motion of a mechanical system is stable if its total mechanical energy decreases all the time. In using the direct method to analyze the stability of a nonlinear system, the idea is to construct a scalar energy-like function (a Lyapunov function) for the system, and to see whether it decreases. The power of this method comes from its generality: it is applicable to all kinds of control systems, be they time-varying or time-invariant, finite dimensional or infinite dimensional. Conversely, the limitation of the method lies in the fact that it is often difficult to find a Lyapunov function for a given system. (Direct excerpt from [SL91, RV13])

On the other hand, the optimal controls can also be obtained as the solution of the Hamilton Jacobi Bellman (HJB) equation. The HJB equation is a nonlinear partial differential equation and one must resort to approximate numerical schemes for its solution. Numerical schemes typically discretize the state-space; hence, the 1resulting problem size grows exponentially with the dimension of the state-space. [RV13]

AA203 discusses both methods in greater detail. In AA274A, we will look into an implementation example of closed-loop control law based on Lyapunov stability analysis in the context of pose stabilization.

4.3.1 Pose stabilization

Stabilization of nonlinear systems is one of the most important, extensively studied problems in control theory. [RV13] Consider a differential drive mobile robot with kinematics

$$\begin{aligned} \dot{x}(t) &= V(t)cos(\theta(t)) \\ \dot{y}(t) &= V(t)sin(\theta(t)) \\ \dot{\theta}(t) &= \omega(t). \end{aligned} \tag{12}$$

We define coordinates of the robot as shown in Figure 1. The control inputs of this system are V and ω.

We cast the pose stabilization problem on polar coordinates and set of kinematic equations accordingly. First, We define ρ, α and δ as the following:

- ρ: distance from the robot to the goal

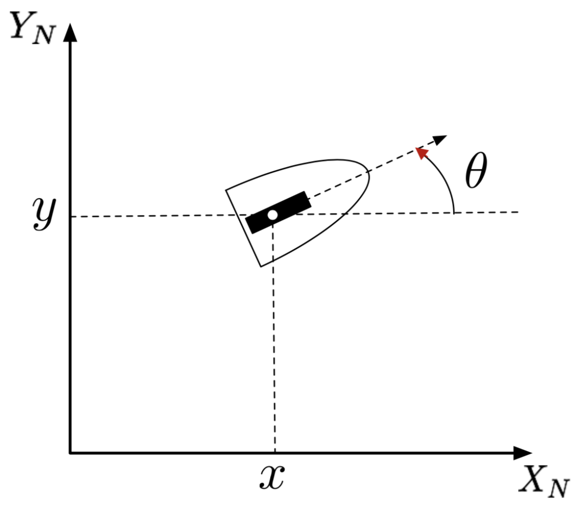

Figure 1: Pose stabilization problem in Cartesian coordinates

- α: bearing error, i.e., angle difference between the goal and current bearing
- δ: angle of the robot with respect to the goal

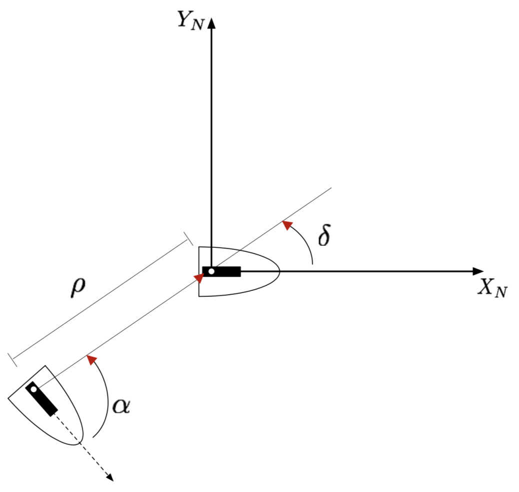

Figure 2: Pose stabilization problem in polar oordinates

Geometric relationships between $(\rho, \alpha, \delta) and$ and (X, Y, θ) coordinates are:

$$\begin{aligned} \rho &= \sqrt{x^2 + y^2} \\ \alpha &= \text{atan2}(y, x) - \theta + \pi \\ \delta &= \alpha + \theta. \end{aligned} \tag{13}$$

Next we derive kinematic equations in polar coordinates. Let P be the current position of the robot, and O, the origin. $\overrightarrow{OP}$ indicates the position vector of our robot. The control input ω cannot move the position, so $\dot{\rho}$ comes from the other input V.

$$\begin{aligned} \dot{\rho} &= \vec{V} \cdot \overrightarrow{OP} / ||\overrightarrow{OP}|| \\ \dot{\rho} &= V \cos(\pi - \alpha) = -V \cos\alpha \end{aligned} \tag{14}$$

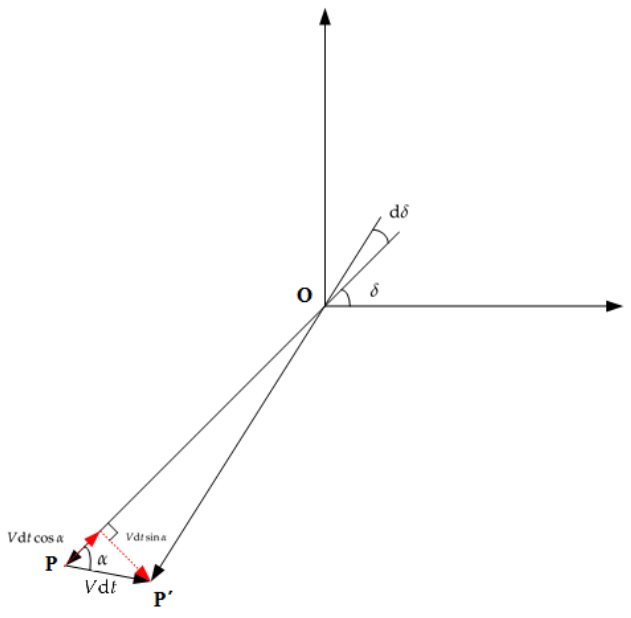

Figure 3: Polar Coordinates

$\dot{\delta}$ is only related to the position of the vehicle. Consider that the vehicle moves for an infinitesimal time period dt. The displacement of the vehicle is Vdt. From Figure 3, we can see that we can also express this displacement as ρdδ, so:

$$\begin{aligned} \rho \mathrm{d}\delta &= V \mathrm{d}t \sin\alpha \\ \dot{\delta} = \frac{\mathrm{d}\delta}{\mathrm{d}t} &= \frac{V\sin\alpha}{\rho} \end{aligned} \tag{15}$$

The expression for $\dot{\alpha}$ can be directly derived from (12), (13) and (15):

$$\begin{aligned} \dot{\alpha} &= \dot{\delta} - \dot{\theta} \\ \dot{\alpha} = \frac{\mathrm{d}\delta}{\mathrm{d}t} &= \frac{V\sin\alpha}{\rho} - \omega \end{aligned} \tag{16}$$

From (14), (15) and (16) we arrive at the kinematic equations of the robot in polar coordinates.

$$\begin{aligned}\dot{\rho}(t) &= -V(t)\cos(\alpha(t))\\ \dot{\alpha}(t) &= \frac{V(t)\sin(\alpha(t))}{\rho(t)} - \omega(t)\\ \dot{\delta}(t) &= \frac{V(t)\sin(\alpha(t))}{\rho(t)}\end{aligned} \tag{17}$$

In order to achieve the goal posture, the variables (ρ, α, θ) should all converge to zero.

In terms of these variables, we can construct a Lyapunov function:

$$V(\rho,\alpha,\theta) = \frac{1}{2}\lambda\rho^2 + \frac{1}{2}(\alpha^2 + k_3\delta^2) \tag{18}$$

For global asymptotic stability we require that the time derivative of this Lyapunov function is always negative. Following the derivation in [ACBB95], it can be shows that this holds if we make the following choices for the controls V, ω:

$$\begin{aligned}V &= k_1 cos(\alpha)\\ \omega &= k_2\alpha + k_1\frac{sin(\alpha)cos(\alpha)}{\alpha}(\alpha + k_3\delta),\end{aligned} \tag{19}$$

where $k_1, k_2, k_3 > 0$.

In Homework 1, you will have a chance to implement the closed-loop control law based on the Lyapunov function, and see it for yourself that our robot indeed successfully “parks” at the goal posture.

References

[201] Mathworks 2019. Pure pursuit controller - matlab & simulink. `https://www.mathworks.com/help/robotics/ug/pure-pursuit-controller.html`. (Accessed on 10/04/2019).

[ACBB95] Michele Aicardi, Giuseppe Casalino, Antonio Bicchi, and Aldo Balestrino. Closed loop steering of unicycle like vehicles via lyapunov techniques. *IEEE Robotics & Automation Magazine*, 2(1):27–35, 1995.

[Lev09] Jean Levine. *Analysis and Control of Nonlinear Systems: A Flatness-based Approach (Mathematical Engineering)*. Springer, 2009.

[Mur] Richard M Murray.

[RPK13] Klaus Röbenack, Fabian Paschke, and Carsten Knoll. Nonlinear control with approximately linear tracking error. In *2013 European Control Conference (ECC)*, pages 149–154. IEEE, 2013.

[RV13] Arvind Raghunathan and Umesh Vaidya. Optimal stabilization using lyapunov measures. *IEEE Transactions on Automatic Control*, 59(5):1316–1321, 2013.

[SL91] Jean-Jacques Slotine and Weiping Li. *Applied Nonlinear Control*. Pearson, 1991.