Principles of Robot Autonomy I

Trajectory tracking

Attendance Form

AA 174A | Lecture 3

Agenda

- Trajectory tracking
  - Based on differential flatness techniques
  - Based on LQR techniques

- Readings
  - Chapter 3, sections 3.1 – 3.3 in D. Gammelli, J. Lorenzetti, K. Luo, G. Zardini, M. Pavone. *Principles of Robot Autonomy*. 2026.
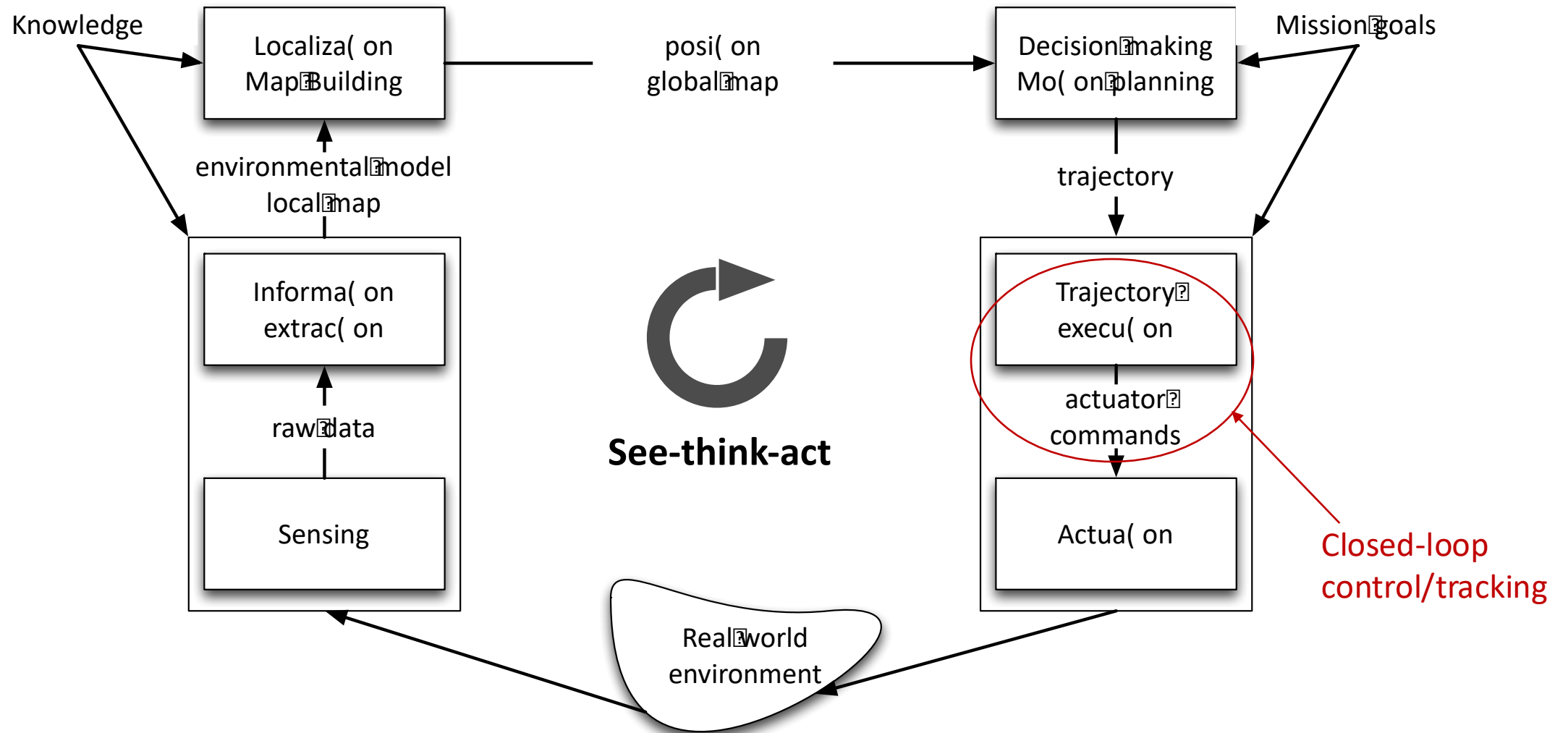
Trajectory tracking

- Back to two-step design strategy

Tracking control law

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

- Reference trajectory and control history (i.e., $\mathbf{x}_d(t)$ and $\mathbf{u}_d(t)$) are computed via open-loop techniques (e.g., differential flatness)

- For reference tracking (Problem 2(v-vi) in pset 1)
  - Geometric (e.g., pursuit) strategies
  - Linearization (either approximate or exact) + linear structure
  - Non-linear control
  - Optimization-based techniques (e.g., MPC)

The see-think-act cycle

Knowledge

Localiza(on
Map Building

posi(on
global map

Decision making
Mo(on planning

Mission goals

environmental model
local map

Informa(on
extrac(on

trajectory

Trajectory
execu(on

raw data

actuator
commands

Sensing

Actua(on

See-think-act

Closed-loop
control/tracking

Real world
environment

Differential flatness (recap)

- A nonlinear system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ is differentially flat if there exists a set of outputs $\mathbf{z} = \alpha(\mathbf{x}, \mathbf{u}, \ldots, \mathbf{u}^{(p)})$ such that

$$\mathbf{x} = \beta(\mathbf{z}, \dot{\mathbf{z}}, \ldots, \mathbf{z}^{(q)})$$

$$\mathbf{u} = \gamma(\mathbf{z}, \dot{\mathbf{z}}, \ldots, \mathbf{z}^{(q)})$$

- One can then use any interpolation scheme (e.g., piecewise polynomial (spline)) to plan the trajectory of $\mathbf{z}$ in such a way as to satisfy the appropriate boundary conditions

- The evolution of the state variables $\mathbf{x}$, together with the associated control inputs $\mathbf{u}$, can then be computed algebraically from $\mathbf{z}$

Trajectory tracking for differentially flat systems

- Example: dynamically extended unicycle model

$$\dot{x}(t) = V\cos(\theta(t))$$
$$\dot{y}(t) = V\sin(\theta(t))$$
$$\dot{V}(t) = a(t)$$
$$\dot{\theta}(t) = \omega(t)$$

- The system is differentially flat with flat outputs (x, y), in particular

$$\begin{bmatrix} \ddot{x}(t) \\ \ddot{y}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -V\sin(\theta) \\ \sin(\theta) & V\cos(\theta) \end{bmatrix}}_{:=J} \begin{bmatrix} a \\ \omega \end{bmatrix} := \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

Trajectory tracking for differentially flat systems

- Then one can use the following virtual control law for trajectory tracking:

$$w_1 = \ddot{x}_d + k_{px}(x_d - x) + k_{dx}(\dot{x}_d - \dot{x})$$

$$w_2 = \ddot{y}_d + k_{py}(y_d - y) + k_{dy}(\dot{y}_d - \dot{y})$$

where $k_{px}, k_{dx}, k_{py}, k_{dy} > 0$ are control gains

- Such a law guarantees exponential convergence to zero of the Cartesian tracking error

Trajectory tracking for differentially flat systems

- More broadly, suppose system is differentially flat: the full state and control trajectories can be computed from flat outputs $(\mathbf{z}, \dot{\mathbf{z}}, \ldots, \mathbf{z}^{(q)})$

- Define: $\mathbf{z}^{(q+1)} = \mathbf{w}$

- One can then design a tracking controller by using linear control techniques; in particular, for a given reference flat output $\mathbf{z}_d$, define the *component-wise* error

$$e_i := z_i - z_{i,d}, \text{ which implies } e_i^{(q+1)} = w_i - w_{i,d}$$

- For guaranteed convergence to zero of tracking error, one can set

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

with the gains $\{k_{i,j}\}$ chosen so as to enforce stability

LQR-based methods

- The previous approach only works for differentially flat systems

- How can we control more general classes of systems?
  - Nonlinear control techniques
  - Linear-quadratic regulation (LQR) methods

LQR-based methods

- The previous approach only works for differentially flat systems

- How can we control more general classes of systems?
  - Nonlinear control techniques
  - Linear-quadratic regulation (LQR) methods

Linear-quadratic regulator (LQR)

- How can we regulate (i.e., drive to the origin) the linear system $\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$ with minimum control effort?

- We define the optimal control problem

$$\min_{\mathbf{u}} \quad \mathbf{x}(T)'F\mathbf{x}(T) + \int_0^T (\mathbf{x}(t)'Q\mathbf{x}(t) + \mathbf{u}(t)'R\mathbf{u}(t))dt$$

$$\text{s.t.} \quad \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$

- The optimal solution is of the form $\mathbf{u}_t = K_t\mathbf{x}_t$ where $K_t = -R^{-1}B'P_t$ and the matrix P_t solves the continuous time Riccati diff. equation:
$$\dot{P}_t = -A'P_t - P_tA + P_tBR^{-1}B'P_t - Q \text{ with } P_T = F$$

- Note: this results holds even in the more general case $\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)$ (just plug $A(t)$ and $B(t)$ in the Riccati equation)

Tracking LQR – linear case

- Consider the linear system $\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$ and assume we would like to track a reference trajectory $\left(\mathbf{x}_d(t), \mathbf{u}_d(t)\right)_t$

- Define error variables $\delta\mathbf{x}(t) := (\mathbf{x}(t) - \mathbf{x}_d(t))$ and $\delta\mathbf{u}(t) := (\mathbf{u}(t) - \mathbf{u}_d(t))$, which leads to the dynamical system
$$\delta\dot{\mathbf{x}}(t) = A\delta\mathbf{x}(t) + B\delta\mathbf{u}(t)$$

- Define optimal control problem

$$\min_{\mathbf{u}} \quad \delta\mathbf{x}(T)'F\,\delta\mathbf{x}(T) + \int_0^T (\delta\mathbf{x}(t)'Q\,\delta\mathbf{x}(t) + \delta\mathbf{u}(t)'R\,\delta\mathbf{u}(t))dt$$
$$\text{s.t.} \quad \delta\dot{\mathbf{x}}(t) = A\delta\mathbf{x}(t) + B\delta\mathbf{u}(t)$$

- Optimal solution is $\delta\mathbf{u}_t = K_t\,\delta\mathbf{x}_t$ (same K_t as before), which leads to control $\mathbf{u}(t) = \mathbf{u}_d(t) + \delta\mathbf{u}(t)$

Tracking LQR – nonlinear case

- Consider the non-linear system $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$ and assume we would like to track a reference trajectory $\big(\mathbf{x}_d(t), \mathbf{u}_d(t)\big)_t$

- Key idea: make the system "linear" by linearizing around $\big(\mathbf{x}_d(t), \mathbf{u}_d(t)\big)_t$:

$$\dot{\mathbf{x}}(t) \approx \mathbf{f}\big(\mathbf{x}_d(t), \mathbf{u}_d(t)\big) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\big(\mathbf{x}_d(t), \mathbf{u}_d(t)\big)\big(\mathbf{x}(t) - \mathbf{x}_d(t)\big) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\big(\mathbf{x}_d(t), \mathbf{u}_d(t)\big)\big(\mathbf{u}(t) - \mathbf{u}_d(t)\big)$$

$$= \dot{\mathbf{x}}_d(t) + A(t)\big(\mathbf{x}(t) - \mathbf{x}_d(t)\big) + B(t)\big(\mathbf{u}(t) - \mathbf{u}_d(t)\big)$$

- As before, we get a linear system in the error variables:
$$\delta\dot{\mathbf{x}}(t) = A(t)\delta\mathbf{x}(t) + B(t)\delta\mathbf{u}(t)$$

- Optimal solution is $\delta\mathbf{u}_t = K_t\,\delta\mathbf{x}_t$ which leads to control $\mathbf{u}(t) = \mathbf{u}_d(t) + \delta\mathbf{u}(t)$ (where K_t is again obtained from the Riccati eq.)

Transferring results to discrete case

- Same ideas apply for discrete-time systems, i.e., $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$

- Key difference: control is $\mathbf{u}_k = \mathbf{u}_k^d + \delta\mathbf{u}_k$, with $\delta\mathbf{u}_k = K_k\,\delta\mathbf{x}_k$, where $K_k = -(R + B_k'P_{k+1}B_k)^{-1}B_k'P_{k+1}A_k$ and P_k is iteratively obtained from the discrete-time Riccati equation:

  $P_k = A_k'P_{k+1}A_k - A_k'P_{k+1}B_k(R + B_k'P_{k+1}B_k)^{-1}B_k'P_{k+1}A_k + Q$ with $P_N = F$

- Similarly as before:

  - $A_k = \frac{\partial\mathbf{f}}{\partial\mathbf{x}}\left(\mathbf{x}_k^d, \mathbf{u}_k^d\right)$ and $B_k = \frac{\partial\mathbf{f}}{\partial\mathbf{u}}\left(\mathbf{x}_k^d, \mathbf{u}_k^d\right)$

  - R, Q, F have the same interpretation as before, namely tracking penalty, control effort penalty, and final tracking error penalty

Next time