Advanced Robotics

Revision Lecture

Steve Tonneau
School of Informatics
University of Edinburgh

Exam: **MAYBE NOT UP TO DATE!**

- Please check website for the latest info: https://exams.is.ed.ac.uk/

INFR11213: Advanced Robotics (INFR11213)

Venue

McEwan Hall - Foyer Room 1 & 2 (Enter via the Pavilion)

Date: Thursday, 19th December 2024

Time: 2:30 p.m. to 4:30 p.m.

Exam Topics

1. Coordinate Transformations
2. Forward and Inverse Geometry
3. Dynamics
4. Digital System and Digital Controllers (PID)
5. Path & Motion Planning I, II
6. **Optimisation**
7. Tutorials
8. Software Lab
9. Hardware Labs
10. **No** reinforcement learning

Homogeneous Transformation Matrix trick

- ${}^{A}\mathbf{p} = {}^{A}\mathbf{R}_{B}{}^{B}\mathbf{p} + \mathbf{t}$ => annoying to write (especially when composing)

- This operation can be written in matrix form:

$$\begin{bmatrix} {}^{A}\mathbf{p} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} {}^{A}\mathbf{R}_{B} & t \\ \mathbf{0}_3 & 1 \end{bmatrix}}_{{}^{A}\mathbf{M}_{B} \in \mathbb{R}^{4\times 4}} \begin{bmatrix} {}^{B}\mathbf{p} \\ 1 \end{bmatrix}$$

- With

$${}^{B}\mathbf{M}_{A} = ({}^{A}\mathbf{M}_{B})^{-1} = \begin{bmatrix} {}^{B}\mathbf{R}_{A} & -{}^{B}\mathbf{R}_{A}t \\ \mathbf{0}_3 & 1 \end{bmatrix}$$

Special groups for rotations

An element of the group….

Can be represented as …

$$r \in SO(3) \simeq \mathbf{R} \in \mathbb{R}^{3\times 3}$$ Rotation matrix

rotation

$\mathbf{q} \in \mathbb{H} \simeq \mathbb{R}^4, \|\mathbf{q}\| = \mathbf{1}$ quaternion

$\mathbf{w} \in so(3) \simeq \mathbb{R}^3$ A velocity ?

$$m \in SE(3) \simeq \mathbb{R}^3 \times SO(3) \simeq \mathbb{R}^3 \times \mathbb{R}^{3\times 3} \simeq \mathbb{R}^{4\times 4}$$ Homogeneous matrix

displacement

translation rotation

$\mathbb{R}^3 \times \mathbb{H} \simeq \mathbb{R}^7$

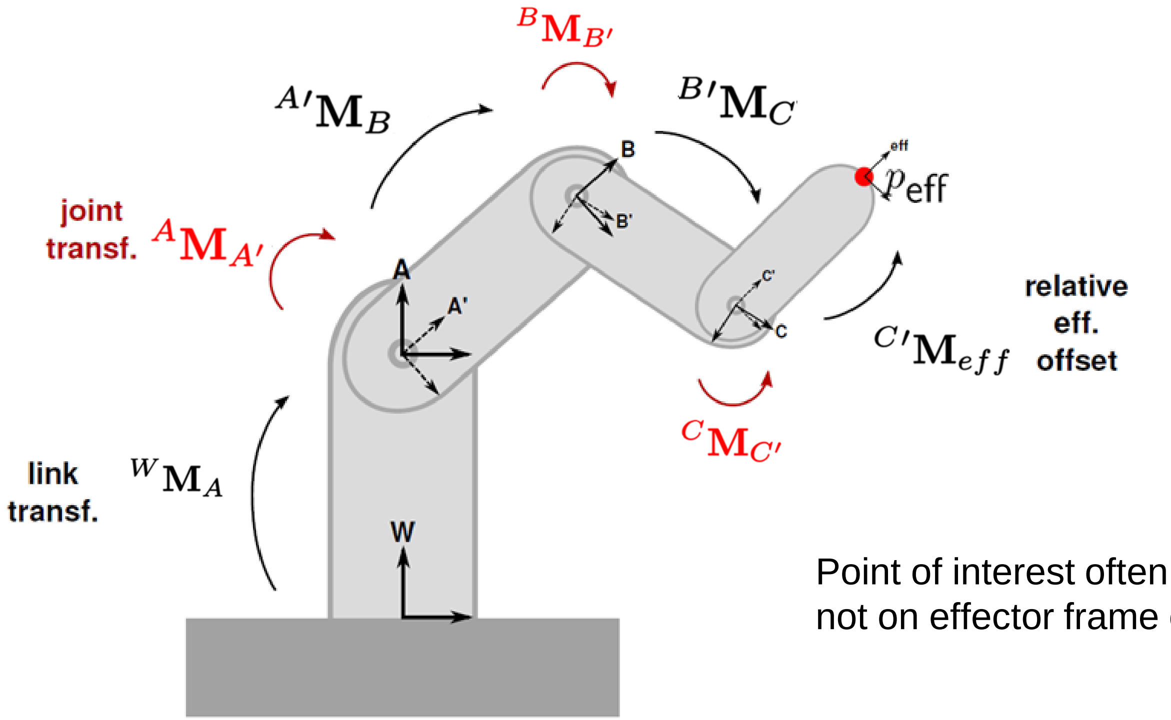
$\mathcal{V} \in se(3) = \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} \simeq \mathbb{R}^6$ A spatial velocity ?

Kinematic chain and map

Generally, frames placed as follows to simplify the variable transformations

Black placements are constant

Red placement are functions of q



Point of interest often
not on effector frame origin

$$^{W}\mathbf{M}_{eff}(\mathbf{q}) = {^{W}\mathbf{M}_{A}}\,{^{A}\mathbf{M}_{A'}(\mathbf{q})}\,{^{A'}\mathbf{M}_{B}}\,{^{B}\mathbf{M}_{B'}(\mathbf{q})}\,{^{B'}\mathbf{M}_{C}}\,{^{C}\mathbf{M}_{C'}(\mathbf{q})}\,{^{C'}\mathbf{M}_{eff}}$$

Forward / inverse kinematics

- Forward **kinematics** consists, given configuration and velocity in configuration space, in computing the velocity of a rigid body in the cartesian space:

$$FK : \mathbf{q}, \mathbf{v}_q \longrightarrow \nu$$

- Inverse **kinematics** consists, given a desired velocity ν^* in the cartesian space (and the current configuration), in computing a velocity in the configuration space result in a velocity as close as possible to ν^*:

$$IK : \nu^*, \mathbf{q} \longrightarrow \mathbf{v}_q$$

Solution to the unconstrained IK problem:

$$v_q{}^* = J^\dagger \nu^*$$

With $J^\dagger$ Moore Penrose pseudo-inverse

However, we could consider additional constraints to our problem: joint limits, velocity limits, etc:

IK with constraints: quadratic programming

- ❑ Velocity bounds (element-wise)

$$\min_{v_q} ||J(q)v_q - \nu^*||^2$$

s.t. $\quad v_q^- \leq v_q \leq v_q^+$

- ❑ Joint bounds (using euler integration over a time step)

$$\mathbf{q}^- \leq \mathbf{q} + \Delta t v_q \leq \mathbf{q}^+$$

- ❑ Can also add other cost functions…

- ❑ ${v_q}^* = J^\dagger \nu^*$ no longer optimal solution

However, easy to solve using a Quadratic Program solver (e.g. quadprog)

Generalising the notion of task

- Not all tasks are just a matter of tracking end-effector trajectories
- Task = a control objective (as in examples at the start of the control lecture)
- A task can be described as a function *e* to minimise error (as in optimal control)
  - Denote e as measuring the **error** between the **real** and **reference** outputs

$$\underbrace{e(\mathbf{x}, \mathbf{u}, t)}_{\text{error}} = \underbrace{y(\mathbf{u}, t)}_{\text{measure}} - \underbrace{y^*(t)}_{\text{reference}}$$

- A large variety of such tasks can then fit into ID control. Relevant ones for your labs are postural tasks (tracking a reference configuration) and force control tasks, e.g. for contact interactions.

IK vs IG

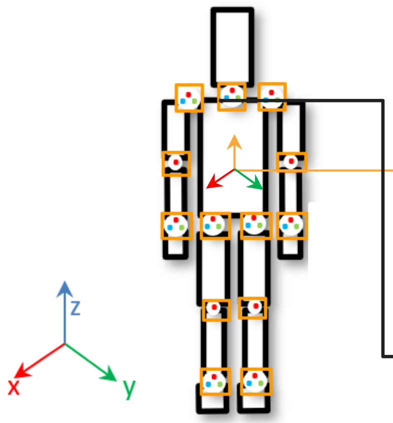
- Inverse kinematics (also called differential IK) is a linear, convex problem, very easy to solve

- Inverse geometry (also called IK) is a non-linear problem, very hard to solve

- When trying to solve IG iteratively, we can use the pseudo-inverse of the jacobian to locally update a configuration towards one that is closer to the goal. This is similar to performing one step of gradient descent (See example after)

The configuration space (Lozano-Peréz 83)

- Robot posture is a point **q** in the configuration space C, of dimension n, or n+6 if root is free (free-flyer joint), with n number of internal Degrees Of Freedom (dof)
  - Each internal dof represented by a joint parameter, subset of **q**
  - If using quaternions to represent free-flyer rotation, **q** is ***represented*** with n+7 != n+6 variables



$$\mathbf{q} = \begin{bmatrix} x \\ y \\ z \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ \alpha \\ \beta \\ \gamma \\ \cdots \end{bmatrix} \in \mathbb{R}^{n+7}$$

q is used to describe both a quaternion and a configuration in the littérature …

3D environment W

Configuration space C

2 manifolds (subsets) for C

- Given a point q in C, using Forward Geometry we can determine whether:
  - q is in collision (in C_{free}) => p(q) = true
  - q is not in collision (in C_{obs}) => p(q) = false
- Given:
  - a current configuration q_c
  - a goal configuration q_g
- Design an algorithm to compute a collision free path from q_c to q_g

Sampling based motion planning summary

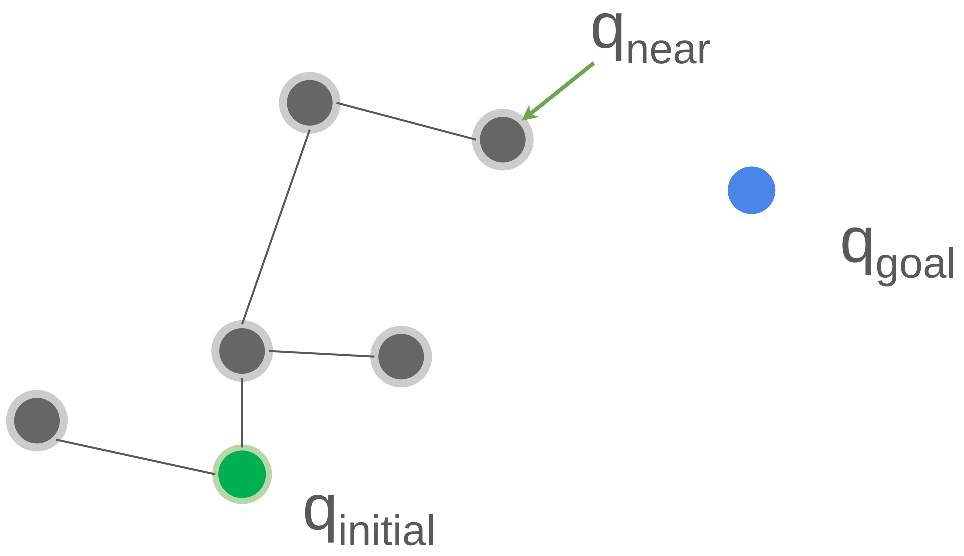
- We have (hopefully) come up with the principles for a global planning algorithm
- A sampling based motion planning algorithm generates a graph were:
  - Nodes are points in the feasible space (in our case C_{free})
  - Edges are feasible paths between Nodes computing with a **local steering method**:
    - In geometric case, often obtained by interpolation
    - **Can be as complex as required by the considered problem**
- The formulation is very generic and can be used to represent any robotics planning problem (RRTs were developed for vehicle control, ie differential constraints)

Basic RRT algorithm – single query variant

Pseudo code

Algorithm 1 BUILD_RRT(q_{init})

```
T.init(q_init);
for k = 1 to K do
    q_rand ← RANDOM_CONFIG();
    q_near ← NEAREST_NEIGHBOR(q_rand, T);
    if edge_valid(q_rand, q_near) then
        T.add_vertex(q_rand);
        T.add_edge(q_near, q_rand);
        if close_to_goal(q_rand) then
            return SUCCESS
return FAILURE
```



Rigid body dynamics equations

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) = \tau$$

C is a vector with Coriolis plus centrifugal terms

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau$$

C is a matrix with Coriolis plus centrifugal terms

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \tau$$

Joint Space Method

Choose a desired acceleration $\ddot{q}_t^*$ that implies a *PD-like behavior around the reference trajectory*!

$$\ddot{q}_t^* = \ddot{q}_t^{\text{ref}} + K_p(q_t^{\text{ref}} - q_t) + K_d(\dot{q}_t^{\text{ref}} - \dot{q}_t)$$

$$M(q)\,\ddot{q}^* + F(q,\dot{q}) = u^*$$

This is a standard and convenient way of tracking a reference trajectory when the **robot dynamics are known**: *all the joints will behave exactly like a 1D point mass around the reference trajectory!*

Inverse dynamics control in a nutshell

- Given $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$, compute torque commands τ that achieve desired acceleration $\ddot{\mathbf{q}}^d$.
- Given a reference $\mathbf{q}^r(t)$ find $\tau(t)$ such that resulting $\mathbf{q}(\tau(t))$ follows $\mathbf{q}^r(t)$
- We assume we can measure $\mathbf{q}$ and $\dot{\mathbf{q}}$
- We set $\tau = \mathbf{M}\ddot{\mathbf{q}}^d + \mathbf{h}$, and now we must compute desired $\ddot{\mathbf{q}}^d$

$$\underbrace{\ddot{\mathbf{q}}^d = \ddot{\mathbf{q}}^r}_{\ddot{\mathbf{e}}} - \mathbf{K}_p(\underbrace{\mathbf{q} - \mathbf{q}^r}_{\mathbf{e}}) - \mathbf{K}_v(\underbrace{\dot{\mathbf{q}} - \dot{\mathbf{q}}^r}_{\dot{\mathbf{e}}})$$

Simpler control laws for manipulator

$$\tau = \underbrace{-K_d\dot{\mathbf{e}} - K_p\mathbf{e}}_{\text{PD}} + \mathrm{g(q)}$$

gravity torque

Even simpler is PID control:

$$\tau = -K_d\dot{\mathbf{e}} - K_p\mathbf{e} + \int_0^t K_i e(s)ds$$

Where integral replaces gravity compensation

All these control laws are stable. In theory, ID control > PD + gravity > PID

Inverse Dynamics control as optimisation problem

- As for inverse kinematics, we can write a least square problem:

$$(\tau^*, \ddot{q}^*) = \underset{\tau, \ddot{\mathbf{q}}}{\text{argmin}} \ ||\ddot{\mathbf{q}} - \ddot{\mathbf{q}}^d||^2$$

Subject to $\quad \tau = \mathbf{M}\ddot{\mathbf{q}} + \mathbf{h}$

- The optimal solution to this is exactly the ID control law if we set

$$\ddot{\mathbf{q}}^d = \ddot{\mathbf{q}}^r - \mathbf{K}_p(\mathbf{q} - \mathbf{q}^r) - \mathbf{K}_v(\dot{\mathbf{q}} - \dot{\mathbf{q}}^r)$$

- So there may be no real advantage here, but the more general framing is useful for more complex problems

Least Square Problem (LSP) (reminder)

- LSP taxonomy:
  - An L_2 norm cost $||Ax - b||^2$
  - Possibly linear inequality / equality constraints ($Cx <= d$; $D x = x$)
- LSPs are a sub-class of convex Quadratic Problems (QPs) which have:
  - Quadratic cost $x^T H x + h^T x$, with $H >= 0$
  - Possibly linear inequality / equality constraints ($Cx <= d$; $D x = x$)
- LSPs and QPs can be solved **extremely** fast with off-the-shelf software
  => compatible with real-time control loops (~ 1 KHz)

Main advantage of optimisation is constraints

- e.g., adding torque limits is much more straightforward:

$$(\tau^*, \ddot{q}^*) = \underset{\tau, \ddot{\mathbf{q}}}{\textbf{argmin}} \; ||\ddot{\mathbf{q}} - \ddot{\mathbf{q}}^d||^2$$

Subject to

$$\tau = \mathbf{M}\ddot{\mathbf{q}} + \mathbf{h}$$

$$\tau^- \leq \tau \leq \tau^+$$

Main advantage of optimisation is constraints

- Assuming constant aceleration at each time step,

$$\dot{\mathbf{q}}(t + \Delta t) = \dot{\mathbf{q}}(t) + \Delta t \ddot{\mathbf{q}}$$

- Joint velocities constraints:

$$(\tau^*, \ddot{q}^*) = \underset{\tau, \ddot{\mathbf{q}}}{\textbf{argmin}} \ ||\ddot{\mathbf{q}} - \ddot{\mathbf{q}}^d||^2$$

Subject to

$$\tau = \mathbf{M}\ddot{\mathbf{q}} + \mathbf{h}$$

$$\tau^- \leq \tau \leq \tau^+$$

$$\dot{\mathbf{q}}(t)^- \leq \dot{\mathbf{q}}(t) + \Delta t \ddot{\mathbf{q}} \leq \dot{\mathbf{q}}(t)^+$$

Optimal control

$$\min_{X,U} \int_0^T l\big(x(t), u(t)\big)dt + l_T\big(x(T)\big)$$

$$\text{s.t.} \quad \dot{x}(t) = f(x(t),u(t))$$

Path cost

Terminal cost

- X and U are functions of t:

  $X: t \in \Re \rightarrow x(t) \in \Re^{nx}$

  $U: t \in \Re \rightarrow u(t) \in \Re^{nu}$

Make sure you understand both TO labs

- The terminal time T is fixed

Trajectory optimisation (tutorials 6 and 7)