Kinematics

- Kinematics:
  - Given the joint angles, compute the hand position
    \[ \mathbf{x} = \Lambda(\mathbf{q}) \]
- Inverse kinematics:
  - Given the hand position, compute the joint angles to attain that position
    \[ \mathbf{q} = \Lambda^{-1}(\mathbf{x}) \]
- As usual, inverse problems might be troublesome!

Geometrically: closed form solution exists in certain cases

By minimization:

\[ J = \frac{1}{2} \| \mathbf{x} - \Lambda(\mathbf{q}) \|_2 \Rightarrow \mathbf{q} = \arg \min_{\mathbf{q}} J \]

Kinematic redundancy: more joints than constraints

- e.g. a rigid body (hand) in space is described by 6 numbers (position + orientation). A robot (or human) arm might have 7 or more joints (degrees of freedom)

Representing kinematics

- Representing rotations and translations between coordinate frames of reference

\[ \Lambda \mathbf{v} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \end{bmatrix} \mathbf{v} \]

\[ \Lambda = \begin{bmatrix} \mathbf{A} & \mathbf{b} \end{bmatrix} \mathbf{v} = \Lambda \mathbf{b} \quad \mathbf{B} \to \mathbf{A} \]

\[ \mathbf{A} = \begin{bmatrix} \mathbf{a}_{b} & \mathbf{0} \end{bmatrix} \mathbf{v} = \Lambda \mathbf{b} \]

\[ \mathbf{a}_{b} = \Lambda \mathbf{b} \quad \mathbf{b}_{a} = \Lambda^{-1} \mathbf{b} \]

Rotation matrix

\[ \Lambda \mathbf{R}_{b} (\Lambda \mathbf{R}_{b})^T = \mathbf{I} \Leftrightarrow (\Lambda \mathbf{R}_{b})^T = (\Lambda \mathbf{R}_{b})^{-1} = \mathbf{R}_{a} \]

Orthogonal matrix

Example: rotation along the Z axis

\[ \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

Rigid body transformations

\[ \| p(t) - q(t) \| = \| p(0) - q(0) \| = \text{constant} \]

- Given that the object is:
  \[ \mathbf{O} \subset \mathbb{R}^3 \]
- The motion of the body is represented by a family of mappings:
  \[ g(t) : \mathbf{O} \to \mathbb{R}^3 \]
- A rigid displacement of the body is:
  \[ g : \mathbf{O} \to \mathbb{R}^3 \]

Action on points and vectors

\[ g, (\mathbf{v}) = g(\mathbf{q}) - g(\mathbf{p}) \]

Where:

\[ \mathbf{v} = \mathbf{q} - \mathbf{p} \]

Note the difference between points and vectors (although both are represented as 3-tuples of numbers). A vector has magnitude and direction and doesn’t belong to a body (free vector).
Then...

\[ g : \mathbb{R}^3 \to \mathbb{R}^3 \]

is a rigid body transformation if:

1. \( \| g(p) - g(q) \| = \| p - q \| \) for all points \( p, q \in \mathbb{R}^3 \)
2. Length is preserved
3. \( g(v \times w) = g(v) \times g(w) \) for all vectors \( v, w \in \mathbb{R}^3 \)
4. The cross product is preserved
5. The inner product is also preserved, thus:
   \[ v^T w = g(v)^T g(w) \]
   i.e. orthogonal vectors remain orthogonal

Some more requirements

- Right handed coordinate systems:
  \[ z = x \times y \]

- If a coordinate system is attached to a rigid body undergoing rigid motion:
  \( v_1, v_2, v_3 \) attached in \( p \) then by effect of \( g \)
  \( g(v_1), g(v_2), g(v_3) \) are attached in \( g(p) \)

Rotation matrix

\[ R_{ab} = [x_{ab} \mid y_{ab} \mid z_{ab}] \]

\( x_{ab} \) Coordinates of the B's principal axis \( a \) relative to \( A \)
\( A \) is the initial frame, \( B \) is the final frame

\[ R_{ab} \in \mathbb{R}^{3 \times 3}, x_{ab}, y_{ab}, z_{ab} \in \mathbb{R}^3 \]

Then:

1. \( x_a, y_a = 0 \) and so forth...
2. \( RR' = R' R = I \)
3. \( \det R = 1 \) for right-handed coordinate systems

Rotation matrix (planar case)

Example: rotation along the Z axis

\[ \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

SO(3) is a group under matrix multiplication

1. Closure
   \( R_1, R_2 \in SO(3) \Rightarrow R_1 R_2 \in SO(3) \)
2. Identity
   \( I \) is the identity element \( IR = R \forall R \)
3. Inverse
   \[ RR^T = R^T R = I, R^T \in SO(3) \]
4. Associativity
   \( (R_1 R_2) R_3 = R_1 (R_2 R_3) \)

The group of rotations \( SO(3) \)

- The set of 3x3 matrices with these properties is denoted:
  \( SO(3) \) which means Special Orthogonal of size 3

- That is:
  \[ SO(3) = \{ R \in \mathbb{R}^{3 \times 3} : RR^T = I, \det R = +1 \} \]
  \( \text{Orthogonal} \quad \text{Special} \)
More simple rotations

Example: rotation along the Y axis
\[
\begin{bmatrix}
\cos \theta & 0 & \sin \theta \\
0 & 1 & 0 \\
-\sin \theta & 0 & \cos \theta
\end{bmatrix}
\]

Example: rotation along the X axis
\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \theta & -\sin \theta \\
0 & \sin \theta & \cos \theta
\end{bmatrix}
\]

Representing 3D rotations

- Sequences of elementary rotations
  - Euler angles: z, y, z or z, x, z
  - Roll, pitch, yaw angles: z, y, x
  - Vector (axis of rotation) and angle

Roto-translation

- Rotation combined with translation
\[
^A v = ^A R_b ^b v + ^A o_b
\]

Homogeneous representation

- To make things uniform
\[
^A v = \begin{bmatrix}
^A R_b & ^A o_b \\
0 & 1
\end{bmatrix} ^b v
\]

Clearly

\[
^A v = ^A T_b ^b T_c ^c v \quad C \to A
\]

\[
^A R_b \begin{bmatrix}
A o_b \\
0 & 1
\end{bmatrix}^{-1} = \begin{bmatrix}
^A R_b & -^A R_b ^b o_b \\
0 & 1
\end{bmatrix}
\]

\[
^A T_b^{-1} = ^b T_A
\]

- Composition of transforms
- Inverse of a rototranslation

Direct kinematics

\[
^\pi T_1(q_1) \cdots ^\pi T_n(q_n)
\]

\[
(x, y, z) = ^\pi T,(q_1, q_2, q_3, q_4) (0, 0, 0)^T
\]

\[
x = \Lambda(q)
orientation = \tilde{\Lambda}(q)
\]
Conventions

- For placing the reference frames on each link
  - Denavit-Hartenberg

- Many times DH parameters are given for a manipulator (and various useful equations are also given wrt DH convention)

Inverse kinematics

- Direct approach
- Geometric
- Minimization
- Neural network, learning

Inverse kinematics

- Direct approach
  - Try solving:
    \[
    x = NL_x(q_x, q_y, q_z, q_w) \\
    y = NL_y(q_x, q_y, q_z, q_w) \\
    z = NL_z(q_x, q_y, q_z, q_w)
    \]
  - for \( q_x, q_y, q_z, q_w \)

Geometric approach

- For certain manipulator the solution exists in close form
  - Decomposable structures (e.g. translation and rotations can be handled separately)
  - Rotations follow certain rules
  - Many industrial manipulators were designed with inverse kinematics in mind

Minimization

- Find the solution to:
  \[
  J = \frac{1}{2} \| x - \Lambda(q) \|^2 \Rightarrow \arg\min_q J
  \]
- Neural network/learning:
  \[(q, x) \rightarrow \Lambda^{-1}\]
- Approximate the inverse out of a family of functions (NN approach) starting from examples

What about velocity?

- Jacobian matrix

  \[
  \begin{bmatrix}
  \frac{dx}{dt} \\
  \frac{dy}{dt} \\
  \frac{dz}{dt}
  \end{bmatrix} = \begin{bmatrix}
  \frac{dx}{dq_1} & \cdots & \frac{dx}{dq_n} \\
  \frac{dy}{dq_1} & \cdots & \frac{dy}{dq_n} \\
  \frac{dz}{dq_1} & \cdots & \frac{dz}{dq_n}
  \end{bmatrix}
  \frac{dq}{dt}
  \]

  \[
  \frac{dx}{dt} = J(q) \frac{dq}{dt}
  \]
Note on representing velocities

- If $\mathbf{x}$ is:
  $$\mathbf{x} = (x, y, z, \phi, \theta, \psi)$$
- Position + Euler angles
  $$\mathbf{v} = (v_x, v_y, v_z, \dot{\phi}, \dot{\theta}, \dot{\psi})$$
- Euler angles derivatives do not have any clear physical meaning
  $$\mathbf{v} = (v_x, v_y, v_z, \omega)$$
- Angular velocity (rate of rotation along the axis)

Anyway...

- Just make sure the representation and the equations are consistent
  $$\mathbf{v} = (v_x, v_y, v_z, \dot{\phi}, \dot{\theta}, \dot{\psi}) \Rightarrow J_x$$
  $$\mathbf{v} = (v_x, v_y, v_z, \omega) \Rightarrow J_x$$

Jacobian

- Formula
  Given the DH representation of transformations
  Considering only rotational joints
  $$J_x = \begin{bmatrix} J_1 & J_2 & \cdots & J_n \end{bmatrix}$$ for $n$ joints
  $$J_i = \begin{vmatrix} z \times p_{e_i} & 0 & 0 & 0 & 0 & 0 \\ a_{e_i} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{vmatrix}$$

Having written

$$T_0 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$T_i = T_i T_{i-1} \cdots T_1$$

When $J$ is invertible

- Can compute the joint velocities to obtain a certain hand velocity
  $$\dot{\mathbf{q}} = J^{-1} \dot{\mathbf{x}}$$
- If $n > 6$, redundancy:
  $$\dot{\mathbf{q}} = J' \dot{\mathbf{x}} + (I - J^+ J) \mathbf{k}$$
- $\mathbf{k}$ is a constant vector

Troubles

- Even if $n \leq 6$ there are many situations where $J$ cannot be inverted (singularities)
  - Movement singularities (chain of rotations)
  - $J$ not invertible because certain elements go to zero
Resolved rate controller

\[ s \rightarrow J^{-1} \rightarrow \frac{1}{s} q \rightarrow q \rightarrow \text{Joint controllers} \]
\[ s \rightarrow \text{Sensors} \]

Static
- Relationship between forces and torques
  \[ dx = Jdq \]
  \[ dq^T \tau = dx^T F \]
  \[ dq^T \tau = dq^T J^T F \]
  \[ \tau = J^T F \]
- Imagining the integrals where appropriate

Another idea

\[ \tau = J^T F \]

- Use this equation to design a force controller:
  - Given F compute the torques to drive the joints

Dynamics
- Two methods to derive the equation of motion (differential equations)
  - Newton-Euler
  - Lagrange formalism

Newton-Euler

- Start from:
  \[ F = \frac{d}{dt} (mv) \]
  \[ \tau = \frac{d}{dt} (J\dot{q}) \]

\[ F = \frac{d}{dt} (mv) \]
\[ \tau = \frac{d}{dt} (J\dot{q}) = \omega \times (J\dot{q}) + J\ddot{q} \rightarrow \text{kinematics} \]

Lagrange formulation

- Lagrange equations:
  \[ \sum_{i} P_{i} \frac{d}{dt} \frac{\partial L}{\partial \dot{q}_{i}} - \frac{d}{dt} \frac{\partial L}{\partial q_{i}} = \frac{\partial L}{\partial q_{i}} \]

\[ L = K - P \]

\[ x_{p} = x_{p}(q_{1}, \ldots, q_{n}) \]

External forces (no potential)

\[ K = \frac{1}{2} mv^2 + \frac{1}{2} \omega^T J \omega \]
For a manipulator

- Take the joint angles as variable, write the position $x$ of the links, write down $K$, $P$ and the external forces

\[ \tau = M(q)\ddot{q} + h(q, \dot{q}) \dot{q} + g(q) \]

- External forces (control)
- Inertia (generalized)
- Coriolis, centrifugal effects
- Gravity

Complexity

- Newton-Euler: $o(n)$
- Lagrange: $o(n')$

Estimation

- Kinematics $\rightarrow$ just measure the params
- Dynamics $\rightarrow$ estimate from data

Dynamics

- Direct dynamics:
  \[ \tau(t) \rightarrow q(t) \]
- Simulation (integrate the equations – Runge-Kutta, Euler, etc.)
- Inverse dynamics:
  \[ q(t) \rightarrow \tau(t) \]

Dynamics and control

- Case 1: parameters are such that feedback gain at each joint is $>>$ gravity, Coriolis, centrifugal, disturbances, etc.
- Case 2: feedback in not enough for high-speed, precision, etc. $\rightarrow$ compensation is required

Case 1

- Approx behavior:
  \[ A\ddot{q} + B\dot{q} + k(q - q^*) = 0 \]
- Can design $k$ or a PID controller to make this system behave as desired

Case 2

- Let’s imagine we know all the parameters with a certain precision:
  \[ \tau = M(q)\ddot{q} + h(q, \dot{q}) \dot{q} + g(q) \]
  \[ \tau_{control} = M(q)u + h(q, \dot{q}) \dot{q} + g(q) \]
  \[ M(q)\ddot{q} + h(q, \dot{q}) \dot{q} + g(q) = M(q)u + h(q, \dot{q}) \dot{q} + g(q) \]
  \[ M(q)\ddot{q} = M(q)u \]
  \[ u = \ddot{q} + k_p(q - q) + k_i(q - q) \]
Case 2 (continued)

\[ q = u \]
\[ u = q' + k_j(q' - q) + k_q(q' - q) \]
\[ \dot{q} = q' + k_j(q' - q) + k_q(q' - q) \]
\[ e = q' - q \]
\[ 0 = \dot{e} + k_e e \]

*Appropriate design of the gains can get arbitrary exponential behavior of the error*