TERMINOLOGY

A DEFINITION OF ROBOT

A reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks.

REVOLUTE AND PRISMATIC JOINT

Advantage of Revolute Joint	Advantage of Prismatic Joint
More compact (take smaller space to achieve the same amount of movement; Easy to manufacture; Easy to actuate; Able to change the orientation of end-effector.	Easy to design and calculate the forward and inverse kinematics; Less susceptible to singularities *; Workspace is rectangular;

CONFIGURATION

A complete specification of the location of every point on the manipulator

Zero configuration: Where all joint values are 0.

Configuration Space: the set of all possible configurations considering only joint limits

WORKSPACE

1. The total volume swept out by the **end-effector point** as the robot does all possible motions
2. The Cartesian Space in which the robot moves

Obstacle: Areas of the workspace that the robot should not occupy

Reachable workspace

Non-holonomic constraint *[1]

LINK,JOINT,END-EFFECTOR,BASE

MULTIPLE COORDINATE FRAMES

algebraic manipulation such as addition or subtraction must be between vectors expressed in the same frame or in parallel frames

ROTATION

VECTOR, COORDINATE, FRAME

algebraic manipulation is meaningless unless between free vectors expressed under **the same** or **parallel** frames

ROTATION MATRIX

$$R_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}$$

$$= [x_1^0, y_1^0, z_1^0]$$

R must be what we call a "SO(3)":

Special Orthogonal group of order 3; Each column is a unit vector; Columns are mutually orthogonal; DOF=3

$R^{-1} = R^T$

Basic rotation matrices: $R_{x,\theta}, R_{y,\theta}, R_{z,\theta}$

The three interpretations of R_1^0:

- The orientation of *one frame (1)* with respect to *another frame (0)*
- Coordinate transformation relating a point *p in two frames (1 and 0)*
- Rotate a vector to yield a new vector *in the same frame (0)*

COMPOSITE ROTATION

Successive rotation about a **fixed frame: pre-multiplying**

Successive rotation about **intermediate frame: post-multiplying** (more commonly used in robots)

PARAMETERIZATION

Euler Angles

Convention: $Z(\phi) - Y(\theta) - Z(\psi)$;

Current axis (post-multiply)

$R =$

$$\begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix}$$

(Total 12 axis combinations)

Given rotation matrix R, corresponding angles are:

$$\begin{cases} \phi = Atan2(r_{23}/r_{13}) \\ \theta = Atan2(\sqrt{1 - r_{33}^2}/r_{33}) \\ \psi = Atan2(r_{32}/-r_{31}) \end{cases}$$

or

$$\begin{cases} \phi = Atan2(-r_{23}/-r_{13}) \\ \theta = Atan2(-\sqrt{1 - r_{33}^2}/r_{33}) \\ \psi = Atan2(-r_{32}/r_{31}) \end{cases}$$

Yaw, Pitch Roll Angles

Convention: $X(\phi) - Y(\theta) - Z(\psi)$

Yaw Pitch Roll

Fixed axis (pre-multiply)

$R =$

$$\begin{bmatrix} c_\phi c_\theta & c_\phi s_\theta s_\psi - s_\phi c_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ s_\phi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix}$$

Axis/Angle Representation

Rotation θ about axis k

$R =$

$$\left[\begin{array}{c|c|c} k_x^2 v_\theta + c_\theta & k_x k_y v_\theta - k_z s_\theta & k_x k_z v_\theta + k_y s_\theta \\ k_x k_y v_\theta + k_z s_\theta & k_y^2 v_\theta + c_\theta & k_y k_z v_\theta - k_x s_\theta \\ k_x k_z v_\theta - k_y s_\theta & k_y k_z v_\theta + k_x s_\theta & k_z^2 v_\theta + c_\theta \end{array}\right]$$

where $v_\theta = 1 - \cos\theta$

Given rotation matrix R, corresponding k and θ are:

$$\theta = \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

$$k = \frac{1}{2\sin\theta} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

Note that k and θ are defined in frame 0. Despite that k has the same coordinates in both frames.

ROTATION MATRIX VS PARAMETER REPRESENTATION

Advantage of Rotation Matrix
Easy for computing and visualizing the orientation after rotation
Disadvantage of Rotation Matrix
Require more space to store; Susceptible to numerical rounding, which may make its determinant is non-zero.

HOMOGENEOUS TRANSFORMATION

$$rigid\ motion = pure\ translation + pure\ rotation$$

$$\mathbf{H} = \begin{array}{c} \begin{array}{cc} \mathbf{R}_1^0 & \quad\quad \mathbf{d}_1^0 \end{array} \\ \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

normal, sliding and approach

SUCCESSIVE TRANSFORMATION

Like in rotation, pre-multiply when relative to fixed/world frame; post-multiply when relative to intermediate frame.

INVERSE TRANSFORM

$$\mathbf{H}_0^1 = \begin{bmatrix} \mathbf{R}_0^1 & \mathbf{d}_0^1 \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} (\mathbf{R}_1^0)^\top & -(\mathbf{R}_1^0)^\top \mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\mathbf{v}_p^0 = \mathbf{R}_1^0 \mathbf{v}_p^1 + \mathbf{d}_1^0 = \mathbf{H}_1^0\, \mathbf{v}_p^1$$

FORWARD KINEMATICS

Objectives: given joint coordinates, what are the task coordinates (state of end effector)?

KINEMATIC CHAIN

Joint i controls link i

Joint i interconnects link i and link $i-1$

D-H PARAMETERS

1. Locate joint axes $z_0, \dots, z_{n-1}$
2. Establish base frame 0

For $i = 1\ to\ n-1$, do:

3. For $i = 1\ to\ n-1$, do:
   Find an x_i which is **perpendicular to and intersects z_{i-1}**(the two D-H rules) and establish frame i.
4. Establish end-effector frame n.
5. Create a table of link parameters:

a_i	distance between z_{i-1} and z_i along x_i
α_i	angle between z_{i-1} and z_i measured in the plane normed to x_i
d_i	distance between x_{i-1} and x_i along z_{i-1}
θ_i	angle between x_{i-1} and x_i measured in the plane normed to z_{i-1}

6. Form homogeneous transformation matrices

$$A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}$$

$$\begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7. $T_n^0 = A_1 \cdots A_n$

TRAJECTORY

Trajectory: a function of time

Common trajectories:

- linear trajectory
- cubic polynomial trajectory
  *[2] the only situation in which there is no solution: $t_s = t_f$, $(\det(mat) = (t_s - t_f)^4 = 0)$
- quintic polynomial trajectory
- central portion linear segments with parabolic blends (LSPB)
  - parabola+line+parabola
  - a special case: Minimum

Time Trajectories / Bang-Bang Trajectories

- All trajectories mentioned above except linear can leave the interval between start and end position during time span.

- Reason why we use linear/cubic instead of quintic: constant velocity (linear); robot is sufficiently rigid (no jerk); lower computational complexity; lower memory usage; limited maximum speed;

- Reason why we usually use odd-order polynomials: they have an even number of coefficients, and usually the number of constraints is an even number!

Common constrains:

	Initial Conditions	Final Conditions
Position	$q(t_0) = q_0$	$q(t_f) = q_f$
Velocity	$\dot{q}(t_0) = v_0$	$\dot{q}(t_f) = v_f$
Acceleration	$\ddot{q}(t_0) = \alpha_0$	$\ddot{q}(t_f) = \alpha_f$
Jerk	$\dddot{q}(t_0) \neq \infty$	$\dddot{q}(t_f) \neq \infty$

INVERSE KINEMATICS

Objectives: given task coordinates, what are the joint coordinates?

Or, *if we know the homogeneous transformation between end-effector and base frame, what are the joint coordinates?*

Given a manipulator and an arbitrary homogeneous transformation, the number of solutions is indeterminate.

KINEMATIC DECOPOLING

For robots with a **spherical wrist**

1. Calculate wrist position:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix}$$

position

2. Solve the joint coordinates before the wrist (e.g. $\theta_1, \theta_2, \theta_3$) and calculate R_3^0
3. Calculate writs orientation

$$\mathbf{R} = \mathbf{R}_3^0 \mathbf{R}_6^3$$
$$\mathbf{R}_6^3 = \left(\mathbf{R}_3^0\right)^{-1}\mathbf{R} = \left(\mathbf{R}_3^0\right)^T \mathbf{R}$$

orientation

INVERSE POSITION

$$\begin{cases} algebriac\ approach \\ geometric\ analysis \end{cases}$$

INVERSE ORIENTATION

A spherical wrist can be solved be solving Euler angles (ZYZ)

$$\theta_{end-2} = \phi, \theta_{end-1} = \theta, \theta_{end} = \psi$$

*[1] A non-holonomic system is a system whose state depends on the path taken in order to achieve it.

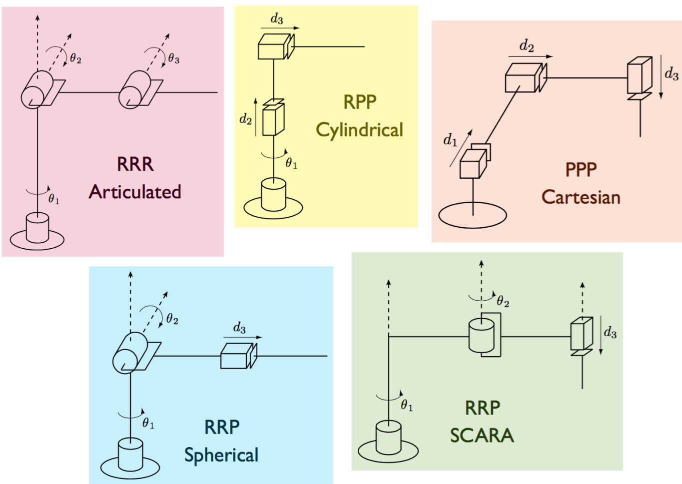
*[2] A mechanical singularity is a position or configuration of a mechanism or a machine where the subsequent behavior cannot be predicted, or the forces or other physical quantities involved become infinite or nondeterministic.

APPENDIX

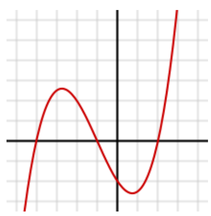
A. Trigonometry Functions

$$\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$$
$$\sin(\alpha+\beta) = \sin\alpha\cos\beta + \sin\beta\cos\alpha$$
$$c^2 = a^2 + b^2 - 2ab\cos\theta$$

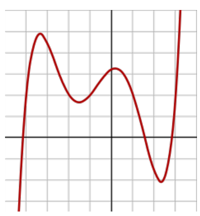
B. Diagrams of Common Manipulators



C. Polynomial Function Plots



cubic



quintic

D. Linear Algebra

1. Inverse matrix
   A $n \times n$ matrix A has an inverse A^{-1} if and only if the rank of A is n, or $det(A) \neq 0$
   $$(AB)^{-1} = B^{-1}A^{-1}$$
2. Inner product (scalar product): $x \cdot y$
   outer product: xy^T
   cross product: $x \times y = (x_2y_3 - x_3y_2)i + (x_3y_1 - x_1y_3)j + (x_1y_2 - x_2y_1)k$

REMINDER

- UNITS: In practice, the elements in homogeneous transformation have units.

QUATERNIONS

UNIT QUATERNION

$$Q = (q_0, q_1, q_2, q_3) = q_0 + iq_1 + jq_2 + kq_3$$

Identity quaternion: $Q_I = (1,0,0,0)$

Conjugate quaternion for Q:

$$Q^* = (q_0, -q_1, -q_2, -q_3)$$

AXIS/ANGLE TO QUATERNIONS

$$axis: \hat{n} = [n_x, n_y, n_z]^T, angle: \theta$$

$$Q = (\cos\left(\frac{\theta}{2}\right), n_x \sin\left(\frac{\theta}{2}\right), n_y \sin(\frac{\theta}{2}), n_z \sin\left(\frac{\theta}{2}\right))$$

ANIMATE ROTATION USING QTN

Multiplying two quaternions:

$$X = (x_0, \vec{x}), Y = (y_0, \vec{y})$$

$$XY = \underbrace{x_0 y_0 - \vec{x}^T \vec{y}}_{z_0} + \underbrace{x_0\vec{y} + y_0\vec{x} + \vec{x}\times\vec{y}}_{\vec{z}}$$

APPLY A UNIT QUATERION'S ROTATION TO A VECTOR

Vector $\vec{v} = (v_x, v_y, v_z)$ in quaternions: $Q_v = (0, v_x, v_y, v_z)$

Rotated $Q_{v'} = QQ_vQ^*$

VELOCITY

NOTATION

$\omega_{i,j}^k$: angular velocity of frame j with respect to frame i expressed in frame k

SKEW-SYMMETRIC MATRICES

Define linear operator $S(\vec{\omega})$:

$$S(\vec{\omega}) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Meaning: $S(\vec{\omega})\vec{r} = \vec{\omega}\times\vec{r} = \vec{v}$

Why we need this?

$$\frac{dR}{d\theta}R^T + R\frac{dR^T}{d\theta} = 0$$

$$define\ S = \frac{dR}{d\theta}R^T, S + S^T = 0$$

$$SR = \frac{dR}{d\theta}$$

Instead of taking differentiation, just form S from angular velocity vector.

APPLICATION

- Calculating the velocity of a point in a rotating frame.

$p^0 = R_1^0 p^1,\ \dot{p}^0 = S(\vec{\omega}(t))R_1^0 p^1$

- Calculating the linear velocity of the end-effector of a robot.

$$p^0 = R_1^0 p^1 + o_1^0$$

$$\dot{p}^0 = S(\vec{\omega}(t))R_1^0 p^1 + \dot{o}_1^0 = \vec{\omega}\times p^0 + \dot{o}_1^0$$

- Understanding how angular velocities combine on a robotic manipulator.

$$R_2^0(t) = R_1^0(t)R_2^1(t)$$

$$\omega_{0,2}^0 = \omega_{0,1}^0 + R_1^0\omega_{1,2}^1$$

JACOBIAN

Jacobian matrix represent how joint velocities turn into end-effector velocities, and it's strongly depends on the robot's current configuration.

$$v_n^0 = J_v\dot{q}, \qquad \omega_n^0 = J_\omega\dot{q}$$

Inverse: $\dot{q} = J_v^{-1}v_n^0$ (only when the Jacobian is non-singular)

USES

- planning and executing smooth trajectories
- determining singular configurations
- executing coordinated anthropomorphic motion
- deriving dynamic equations of motion
- transforming forces and torques from the end-effector to the manipulator joints.

LINEAR JACOBIAN

$$\dot{\vec{p}} = J_v(\vec{q})\dot{\vec{q}}$$

Approach 1:

$$J_v(\vec{q}) = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \end{bmatrix}$$

Approach 2:

For prismatic joint i: $J_{v_i} = \hat{z}_{i-1}$

For revolute joint i:

$$J_{v_i} = \hat{z}_{i-1}\times(o_n - o_{i-1})$$

where z, o are the axis and origins following DH rules and expressed in frame 0.

ANGULAR JACOBIAN

$$\omega = J_\omega(\vec{q})\dot{\vec{q}}$$

For prismatic joint i: $J_{\omega_i} = [0\ 0\ 0]^T$

For revolute joint i:

$$J_{\omega_i} = \hat{z}_{i-1} = R_{i-1}^0\hat{z}$$

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}$$

What do we need to calculate Jacobian?

$$T_n^0 = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ANALYTICAL JACOBIAN*

SINGULARITY

Def: points in the configuration space where infinitesimal motion in a certain direction is not possible and the manipulator loses one or more degrees of freedom (lose rank).

$$Singular \Leftrightarrow \det(J) = 0$$

Find singularity for 6-DOF manipulator with *spherical wrist*: decoupling

$$J = [J_{arm} \mid J_{wrist}] = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}$$

Choose $o_4 = o_5 = o_6$, $J_{12} = 0$

$$\det(J) = \det(J_{11})\det(J_{22})$$

$$J_{22} = [z_3, z_4, z_5]$$

When $z_3 \perp z_4, z_4 \perp z_5, \det(J_{22}) = 0$

FROM FORCE TO TORQUE

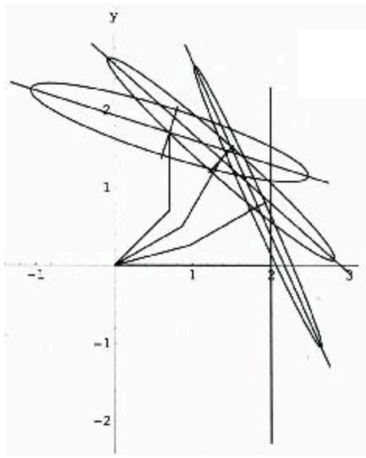
$$\vec{\tau} = J(\vec{q})^T\vec{F}$$

where $F = [F_x, F_y, F_z, n_x, n_y, n_z]$ represent the forces and moments at the end effector.

MANIPULABILITY

$$\xi = J\dot{q}$$

$$\mu = |\det(J)|$$



PATH PLANNING

WORK SPACE VS CONFIGURATION SPACE

configuration space obstacle

ARTIFICIAL POTENTIAL FIELD (in work space)

$$U(q) = U_{att}(q) + U_{rep}(q)$$

Gradient descent $\tau = -\nabla U(q)$

Examples of potential field

conic well potential

$$F_{\text{att},i}(q) = -\zeta_i\,(o_i(q) - o_i(q_f))$$

$$F_{\text{att},i}(q) = -\frac{(o_i(q) - o_i(q_f))}{||(o_i(q) - o_i(q_f))||}$$

parabolic well potential

a common repulsive potential

What is the force on the border of the region of influence?
zero

when $\rho_i(q) > \rho_0$

$$\boxed{F_{\text{rep},i}(q) = 0}$$

when $\rho_i(q) \le \rho_0$

What is the force on the border of the obstacle?
infinity

$$\boxed{F_{\text{rep},i}(q) = \eta_i\left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(o_i(q))}\nabla\rho(o_i(q))}$$

if the obstacle is convex and b is the point on the surface of the obstacle closest to the end-effector

$$\nabla\rho(o_i(q)) = \frac{o_i(q) - b}{||o_i(q) - b||}$$

Procedure

1. $q^0 \leftarrow q_s, i \leftarrow 0$
2. if $\|q^i\text{-}q_f\| > \varepsilon$

$$q^{i+1} \leftarrow q^i + \alpha^i\frac{\tau(q^i)}{\|\tau(q^i)\|}$$

$$i \leftarrow i+1$$

else return

3. go to 2

Apply forces to robot $\vec{\tau} = J_v^T\vec{F}$

CONFIGURATION SPACE MAP

Probabilistic roadmap: Randomly sample configuration space; Determine which pairs of nodes should be connected by a simple path; connect these disjoint components, connect the initial and final configuration to the roadmap, search to find the shortest path in the road map from initial to final configurations

REAL ROBOT

Limitation of Real Robot: mass and extent; gravity; friction and backlash; sensor error; actuator limitation; software speed, etc.

<u>SENSING</u>

Sensors

Potentiometers	Encoder
$V_{out} = \frac{F_1}{R_{all}} V_{in}$	$\theta = \Delta(Q - Q_{zero})$ Δ: $2\pi/4n$ n is the resolution (CyPR) 1 cycle/revolution = 4 counts/revolution
• Cheap and easy • absolute position • wear out • Hard to waterproof or dustproof. • Susceptible to electrical noise • non-negligible friction • finite range	• 2 channels 90 degree out of phase (so that it can tell the direction) • Quadrature decoding gives 4 x finer resolution • relative position (need calibration pose or secondary sensor) • significant noise on velocity

Gear and Capstan

Gear

$$N = \underset{\text{Teeth}}{\frac{n_{\text{out}}}{n_{\text{in}}}} = \underset{\text{Radius}}{\frac{r_{\text{out}}}{r_{\text{in}}}} = \underset{\text{Speed}}{\frac{\omega_{\text{in}}}{\omega_{\text{out}}}} = \underset{\text{Torque}}{\frac{\tau_{\text{out}}}{\tau_{\text{in}}}}$$

Capstan (with the motor) and drum:

$$\theta_{joint} r_{drum} = \theta_{motor} r_{capstan}$$

Advantage of capstan:

- stiff connection with zero backlash
- smooth, efficient, no vibration

Things could cause problem in sensing

- continuous vs discrete
- real value vs integers
- conversion factor
- direction of motion and zero pose

<u>ACTUATION</u>

Motor (spe. DC brush motor)

K_v: *back emf constant* =Torque

$$V(t) = L\frac{di_a}{d_t} + R\,i_a + k_v\,\omega_m$$

constant k_t, where $\tau_m = k_t i_a$

$$\sum \tau \text{ on motor} = \underset{\substack{\text{motor}\\\text{output}\\\text{torque}}}{\tau_m} - \underset{\substack{\text{load}\\\text{torque}\\\text{over g.r.}}}{\tau_l/r} - \underset{\substack{\text{viscous}\\\text{motor}\\\text{friction}}}{B_m\dot{\theta}_m} = (\underset{\substack{\text{motor}\\\text{inertia}}}{J_a} + \overset{\substack{\text{gearhead}\\\text{inertia}}}{J_g})\underset{\substack{\text{motor}\\\text{angular}\\\text{acceleration}}}{\ddot{\theta}_m}$$

Current Amplifier

Use current-drive amplifier instead of voltage-drive amplifier because the motor's output torque is proportional to the current, not the voltage.

$$i_{motor} = i_R = \frac{V_{command}}{R}$$

<u>CONTROL</u>

Stability and transparency

Low-pass filtering (speed smoothing)

$$\underset{\text{new smoothed}}{v_{\text{smooth},k}} = w \cdot \underset{\text{new raw}}{\hat{v}_k} + (1 - w) \cdot \underset{\text{old smoothed}}{v_{\text{smooth},k-1}}$$

small w values smooth a lot | $0 < w < 1$ | large w values don't smooth much

Cutoff frequency: $\lambda = \frac{w}{T(1-w)} \frac{1\ cycle}{2\pi\ rad}$

Proportional feedback controller

$\tau = \kappa(\theta_{des} - \theta)$, κ is proportional gain

$F = k(x_{des} - x)$

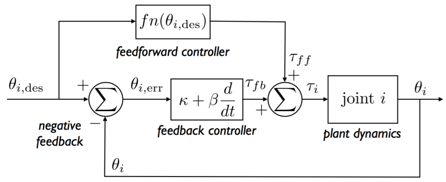
PD controller

Drive the robot to target state.

Feedforward control

Try to compensate for the robot's

$$\tau_m = \underset{\substack{\text{Proportional}\\\text{Gain}}}{k_p}(\underset{\substack{\text{Desired}\\\text{Position}}}{\theta_d} - \underset{\substack{\text{Measured}\\\text{Position}}}{\underline{\theta_m}}) + \underset{\substack{\text{Derivative}\\\text{Gain}}}{k_d}(\underset{\substack{\text{Desired}\\\text{Velocity}}}{\omega_d} - \underset{\substack{\text{Measured}\\\text{Velocity}}}{\underline{\omega_m}})$$



dynamics in advance (e.g. inertial, weight, friction)

HAPTIC ROBOT

<u>ENVIRONMENT</u>

Real; Remote (teleoperation); Virtual

<u>TELEOPERATION CONTROL</u>

Bilateral PD controller

Position scaling

$\vec{x}_{s,des} = \mu\vec{x}_m, \vec{x}_{m,des} = \vec{x}_s/\mu$

don't scale rotational motion

Clutching

$\vec{x}_{s,des} = \mu(\vec{x}_m - (\vec{x}_{clutch}) = \mu\vec{x}_m - \vec{x}_{offset}, \vec{x}_{m,des} = (\vec{x}_s - \vec{x}_{offset})/\mu$

the offset start at zero and accumulate during interaction; usually don't clutch rotation

Rate control

$$\vec{v}_{s,des} = \gamma\vec{x}_m, \vec{F}_m = -k\vec{x}_m$$

used when the workspace is very large; use a centering spring to push the user's hand back to zero.

Make it easier to stop the robot, add dead-band

If $x_m > \delta$: $v_{s,des} = \gamma(x_m - \delta)$

else if $x_m < \delta$ $v_{s,des} = \gamma(x_m + \delta)$

else $v_{s,des} = 0$

APPENDIX

<u>COMPLEX NUMBER</u>

$$i^2 = j^2 = k^2 = -1$$
$$i = jk = -kj$$
$$j = ki = -ik$$
$$k = ij = -ji$$

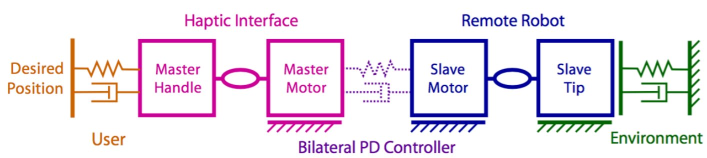
<u>REAL ROBOT PROBLEM ANALYTICS</u>

The code might not execute at even rate. If we assume constant Δt, the robot may have vibration at certain frequency. So you should use time sensor instead.

Singularity: rotational motion (i.e. for spherical wrist) is constrained and may pop/flip about the singularity.

Gravity and weight (two ways to solve, computational and mechanical (counterweight))

<u>DIAGRAM FOR TELEOPERATION</u>



<u>REAL ROBOT FRAMEWORK</u>