Résumé robotique 1

1. Kinematic types and workspace

Industrial robot : ≥ 3 axis manipulator

Manipulator : arrangement of segments and joints

↳ Consist of arm + wrist

Wrist : in general last 3 joints → spherical wrist

2 Spatial description

"pose" : displacement + orientation

"denominate scalar" : 20 cm w/ unit

Standard frames : Base {B} Goal {G} Wrist {W} Tool {T} Station {S} Tool Center Point TCP

Goal : Take T to G

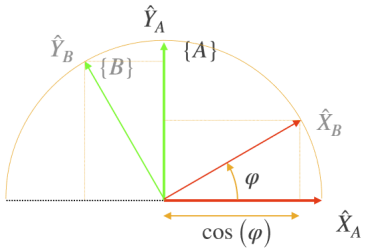
Unit vectors of frame {A} : $\hat{X}_A \ \hat{Y}_A \ \hat{Z}_A$

Orientation representation

Rotation matrix

any point in {B} is seen from {A} as

$$ {}^A p = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} {}^B p $$



$$ {}^A p = {}^A_B R \, {}^B p $$

from B to A — Vue depuis A

$$ {}^A_B R = \begin{bmatrix} {}^A\hat{X}_B & {}^A\hat{Y}_B & {}^A\hat{Z}_B \end{bmatrix} = \begin{bmatrix} \hat{X}_{1B}\cdot\hat{X}_{1A} & \hat{Y}_{1A}\cdot\hat{X}_{1A} & \hat{Z}_{1A}\cdot\hat{X}_{1A} \\ \hat{X}_{2B}\cdot\hat{Y}_{2A} & \hat{Y}_{2B}\cdot\hat{Y}_{2A} & \hat{Z}_{2B}\,\hat{Y}_{2A} \\ \hat{X}_{3B}\cdot\hat{Y}_{3A} & \hat{Y}_{3B}\cdot\hat{Y}_{3A} & \hat{Z}_{3B}\cdot\hat{Y}_{3A} \end{bmatrix} $$

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

$$ {}^A_B R = {}^B_A R^{-1} = {}^B_A R^T \qquad {}^A_C R = {}^A_B R \, {}^B_C R $$

$$ R_X(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{bmatrix} \quad R_Y(\beta) = \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \quad R_Z(\alpha) = \begin{bmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ {}^A_B R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} $$

Euler - Angle convention

Fixed angle (12 sets) : rotation done about original frame, also known as extrinsic rotations

$$ {}^A_B R_{XYZ}(\gamma,\beta,\alpha) = R_z(\alpha)\,R_y(\beta)\,R_x(\gamma) $$

"roll-pitch-yaw"

$$ = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & \overbrace{c\alpha}^{\cos\alpha} s\beta c\gamma + \overbrace{s\alpha}^{\sin\alpha} s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} $$

$$ = {}^A_B R_{Z'Y'X'} = {}^A_{B'}R \, {}^{B'}_{B''}R \, {}^{B''}_{B}R $$

ZYX Euler = XYZ Fixed angle

Euler (12 sets) : rotation done about current moving frame — "intrinsic rotations"

Other typical : ZYZ Euler angle (spherical wrist) = ZY

⚠ Euler-angle has 1 singularity at $\beta = \frac{\pi}{2}$ giving $\arctan2\left(\frac{0}{0},\ldots\right)$ → garbage in C

⇒ occur always on the second axis rotation

A solution at $\beta = \frac{\pi}{2}$: $\gamma = \text{sgn}(\beta)\cdot\text{atan}(r_{12}, r_{22})$, $\alpha = 0$

Pitch

$$ \beta = \arcsin(-r_{31}) $$

Roll

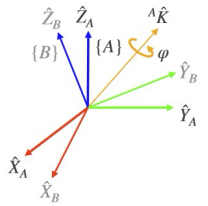
$$ \gamma = \arctan2\left(\frac{r_{32}}{c_\beta}, \frac{r_{33}}{c_\beta}\right) $$

Yaw

$$ \alpha = \arctan2\left(\frac{r_{21}}{c_\beta}, \frac{r_{11}}{c_\beta}\right) $$

Equivalent Angle-Axis

Define a rotation about one special axis with an angle φ



$$ {}^A\hat{K} = \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} \qquad R_K(\varphi) = \begin{bmatrix} k_x k_x v\varphi + c\varphi & k_x k_y v\varphi - k_z s\varphi & k_x k_z v\varphi + k_y s\varphi \\ k_x k_y v\varphi + k_z s\varphi & k_y k_y v\varphi + c\varphi & k_y k_z v\varphi - k_x s\varphi \\ k_x k_z v\varphi - k_y s\varphi & k_y k_z v\varphi + k_x s\varphi & k_z k_z v\varphi + c\varphi \end{bmatrix} $$

- Given a rotation matrix

$$ {}^A_B R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} $$

- The Angle-Axis representation is

⚠ $\sin\varphi = 0$

$$ \varphi = \arccos\left(\frac{r_{11}+r_{22}+r_{33}-1}{2}\right) \qquad {}^A\hat{K} = \frac{1}{2\,s\varphi}\begin{bmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{bmatrix} $$

Always 2 solutions : $({}^A\hat{K}, \varphi)$ $(-{}^A\hat{K}, -\varphi)$

Scaled angle axis representation : $\varphi = |{}^A K|$

φ in rad ${}^A K = \varphi \cdot {}^A\hat{K} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$

Quaternions "sphere of imaginary units"

$Q = q_w + q_x i + q_y j + q_z k$

$i^2 = j^2 = k^2 = ijk = -1$

$r = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$ $e^{r\psi} = \cos\psi + r \cdot \sin\psi$

" q_x, q_y, q_z can be interpreted as a unit vector, while q_w is its rotation by ψ

From equivalent axis:

$$\hat{Q} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \quad q_w = \cos\left(\frac{\psi}{2}\right) \quad q_x = k_x \sin\left(\frac{\psi}{2}\right)$$

$$q_y = k_y \sin\left(\frac{\psi}{2}\right) \quad q_z = k_z \sin\left(\frac{\psi}{2}\right)$$

$$q_x^2 + q_y^2 + q_z^2 + q_z^2 = 1$$

Equivalent rotation:

$$R_q = \begin{bmatrix} 1 - 2q_y^2 - 2q_z^2 & 2(q_x q_y - q_z q_w) & 2(q_x q_z + q_y q_w) \\ 2(q_x q_y + q_z q_w) & 1 - 2q_x^2 - 2q_z^2 & 2(q_y q_z - q_x q_w) \\ 2(q_x q_z - q_y q_w) & 2(q_y q_z + q_x q_w) & 1 - 2q_x^2 - 2q_y^2 \end{bmatrix}$$

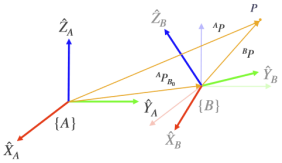
Mapping a point P from {B} to {A}

$$^A p = {}^A_B Q \; {}^B p \; {}^A_B Q^{-1} \qquad Q^{-1} = \begin{bmatrix} q_w \\ -q_x \\ -q_y \\ -q_z \end{bmatrix}$$

Frame translation

General transformation:

$$^A p = {}^A p_{B_0} + {}^A_B R \; {}^B p$$



Translation + rotation (Homogeneity):

$$\begin{bmatrix} ^A p \\ 1 \end{bmatrix} = \left[\begin{array}{ccc:c} & {}^A_B R & & {}^A p_{B_0} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} ^B p \\ 1 \end{bmatrix}$$

$$^A p = {}^A_B T \; {}^B p$$

Compound: $^A_C T = {}^A_B T \; {}^B_C T$

$$^B_A T = {}^A_B T^{-1} = \begin{bmatrix} {}^A_B R^T & -{}^A_B R^T \; {}^A p_{B_0} \\ 0 \; 0 \; 0 & 1 \end{bmatrix}$$

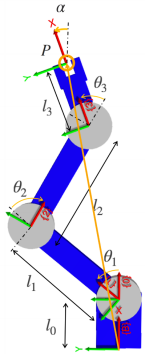
Goal relative to tool: $^T_G T = {}^B_T T^{-1} \; {}^B_S T \; {}^S_G T$

3. Kinematics science of motion (no forces)

Forward find tool pose from joints position

$$\Theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \qquad {}^B X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad X = L(\Theta)$$

Joints positions — Tool pose relative to {B}



1. Draw reference frames at each joints
2. Add frame {0} (first joint nominal)
3. Write $^B_T T = {}^B_0 T \; {}^0_1 T \; {}^1_2 T \; {}^2_3 T \; {}^3_T T^{pose}$
4. Simplify matrice with R and D operators
   $R_z(\theta_4)$ $R_y(\theta_5)$ $D_z(l_0)$ → translation
5. Write out individual T matrices

Denavit-Hartenberg parameters

Inverse find joints value from tool pose

$$\Theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \qquad {}^B X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \boxed{\Theta = L^{-1}(X)}$$

$$\begin{bmatrix} 1 & 0 & 0 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

3 ways:

① Given $^B_T T$, find $^0_W T$

$$^0_W T = {}^B_0 T^{-1} \; {}^B_T T \; {}^W_T T^{-1}$$

"remove offsets"

⇒ we have $x_w = \ldots$ $y_w = \ldots$ $\alpha = \ldots$

Compute $^0_W T = {}^0_1 T \; {}^1_2 T \; {}^2_W T$

3 equations with 3 unknowns → solve algebraicly

② Geometric solution using cos theorem

③ Iterative solution knowing forward kin and jacobian matrix

⚠ Almost always multiple solutions
→ Choose one with - minimum travel
- less energy
- fastest motion

⚠ joint limits, outside workspace

Spherical wrist inverse kinematic

$\theta_5 = \pm \arccos(r_{33})$ — 2 solutions

$\theta_4 = \arctan\left(\frac{r_{23}}{s_5}, \frac{r_{13}}{s_5}\right)$

$\theta_6 = \arctan\left(\frac{r_{32}}{s_5}, \frac{r_{31}}{s_5}\right)$

Jacobian matrix

Linear approximation of how a function changes in each direction.
→ has a reference frame

$${}^B X = L(\Theta) \quad \delta {}^B X = {}^B J(\theta)\delta\theta \quad {}^B J(\Theta) = \frac{\delta L(\Theta)}{\delta \Theta}$$

$$ {}_B\begin{bmatrix} \delta x \\ \delta y \\ \delta \varphi \end{bmatrix} = {}^B J(\Theta) \begin{bmatrix} \delta\theta_1 \\ \delta\theta_2 \\ \delta\theta_3 \end{bmatrix} \quad 3\times 3 $$

$$\delta x = \frac{\partial x}{\partial \theta_1}\delta\theta_1 + \frac{\partial x}{\partial \theta_2}\delta\theta_2 + \frac{\partial x}{\partial \theta_3}\delta\theta_3$$
$$\delta y = \frac{\partial y}{\partial \theta_1}\delta\theta_1 + \frac{\partial y}{\partial \theta_2}\delta\theta_2 + \frac{\partial y}{\partial \theta_3}\delta\theta_3$$
$$\delta \varphi = \frac{\partial \varphi}{\partial \theta_1}\delta\theta_1 + \frac{\partial \varphi}{\partial \theta_2}\delta\theta_2 + \frac{\partial \varphi}{\partial \theta_3}\delta\theta_3$$

Inverse Instantaneous

$${}^B\dot{X} = \begin{bmatrix} {}^B v \\ {}^B \omega \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\gamma} \\ \dot{\beta} \\ \dot{\alpha} \end{bmatrix} = {}^B J(\Theta)\dot{\Theta}$$

$${}^A J(\Theta) = \begin{bmatrix} {}^A_B R & 0 \\ 0 & {}^A_B R \end{bmatrix} {}^B J(\Theta) \quad 6\times 6$$

$$\boxed{\dot{\Theta} = {}^B J^{-1}(\Theta)\, {}^B\dot{X}}$$

Singularities happens when det of Jacobian matrix = 0

often when 2 axis align

Redundant Robots

Joint DOF > cartesian DOF

⇒ Jacobian is not square → no inverse

Pseudo inverse: $J^* = J^T (JJ^T)^{-1}$

$$\boxed{\dot{\Theta} = {}^B J^*(\Theta)\, {}^B\dot{X}}$$

⇒ uses minimum-norm constraint $(\min \|{}^B\dot{X} - J\dot{\theta}\|^2)$

Weighted PI: $J^* = W^{-1} J^T (J W^{-1} J^T)^{-1}$
⇒ minimize cost function (i.e for heavy arms)

Jacobian Null space

Infinite number of solutions

⇒ $\dot{\Theta} = {}^B J^*(\Theta)\, {}^B\dot{X} + N\tilde{\dot{\Theta}}_0$ Null vector if dim(Ker) = 1

${}^B J(\Theta) N = 0 \quad N = I - J^* J$

Static forces

end effector force?

$$\mathcal{F} = \begin{bmatrix} f_x \\ f_y \\ f_z \\ n_x \\ n_y \\ n_z \end{bmatrix} \quad \tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix}$$

$$\boxed{\tau = {}^B J^T(\Theta)\, {}^B\mathcal{F}}$$

↳ Cartesian space to Joint space without Inv Kin ☺

4. Trajectory generation

Defined path → Robot joint trajectories (time history of position, speed, accel.)
Motions often defined in {T} relative to {S}
→ decoupling if robot or tool changes.

Goal: ${}^S_T T_0 \rightarrow {}^S_T T_f$ poses

We specify: poses (points), time, speed, (Tolerance)

Via points (intermediate poses)

Smooth motions ≙ continuous derivatives

Motion State & equations

Defined in an array: $s_X = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$

Particle at x_0 with $\dot{x}_0$

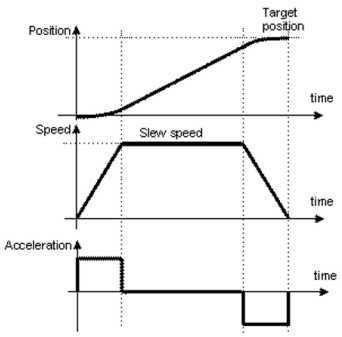
velocity: $\dot{x} = \int \ddot{x} dt = \dot{x}_0 + \ddot{x} t$

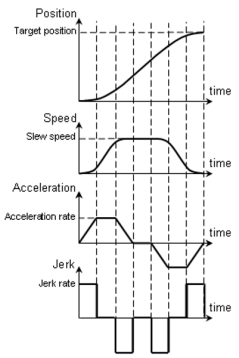
distance: $x = \iint \ddot{x} dt = x_0 + \dot{x}_0 t + \frac{\ddot{x} t^2}{2}$

discrete time: $\theta_{k+1} = \theta_k + \dot{\theta}_k \Delta t + \ddot{\theta}_k \frac{\Delta t^2}{2}$

Motion profiles

In general trapezoidal profiles of 2nd or 3rd order.





2nd order: limit $\dot{x}_{max}$ and $\ddot{x}_{max}$, jerk infinite

3rd order: limit also $\dddot{x}_{max}$ → profile has 7 segments

Motion planner

Interpolate smoothly between 2 poses with motion states

Tasks: increment position and speed states

2nd order: $x = x_0 + \dot{x}_0 t + \frac{\ddot{x} t^2}{2}$

3rd order: $x = x_0 + \dot{x}_0 t + \frac{\ddot{x} t^2}{2} + \frac{\dddot{x} t^3}{6}$ (accel not const)

Update speed based on position

desired

1. Calculate distance to arrival $\boxed{\Delta x = x_d - x}$

$\dot{x} = 0$ and $\Delta x > 0$ → accelerate

$\dot{x} \neq 0 < \dot{x}_{max}$ → accelerate

$\dot{x}_{max}$ → coast

$\Delta x < \Delta x_{stop}$ → decelerate

2. Calculate distance to stop

time to stop: $t_s = \left| \frac{\dot{x}}{\ddot{x}_{max}} \right|$

distance traveled: $x_s = \dot{x} t_s + \frac{\ddot{x}_{max} t_s^2}{2}$ $\quad x_s = \frac{\dot{x}^2}{2 \cdot \ddot{x}_{max}}$ better

$sign(\ddot{x}_{max}) \triangleq -sign(\dot{x})$

3. Set desired speed $\quad \dot{x}_d = \begin{cases} 0, & |\Delta x| \leq x_s \\ \frac{\Delta x}{t}, & |\Delta x| > x_s \end{cases}$

Update acceleration and actual speed, calc. x

1. Update acceleration to reach $\dot{x}_d$

i. Saturate speed (check max) $\dot{x}_d = \dot{x}_d \lceil_{\dot{x}_{max}}$

ii Desired speed change $\Delta \dot{x} = \dot{x}_d - \dot{x}$

iii Desired accel $\ddot{x}_d = \frac{\Delta \dot{x}}{T}$

iv Saturate and set accel

2. Update speed $\dot{x} += \ddot{x} T$ (T: time step)

3. Saturate speed

4. Update position $x += \dot{x} T + \frac{\ddot{x} T^2}{2}$

Blending

Add radius to the path on a via point

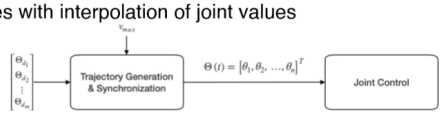
→ keep constant speed

→ each robot manufacturer has its functions

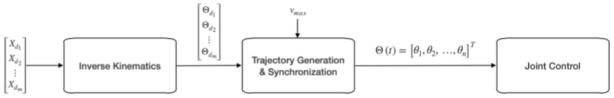
Joint-Space Trajectories

Desired pose $[{}^S_T T_0 \dots {}^S_T T_n]$ → inverse Kin → array of joints values → use 1D trajectory gen.

▸ Motion to desired joint values with interpolation of joint values



▸ Motion to Cartesian pose with interpolation of joint values



The motion planner increment each joint seperately !

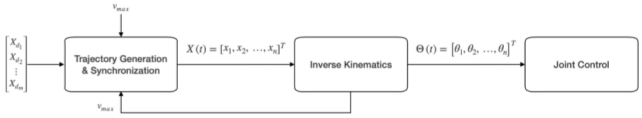
+ Fast execution, - traj. not straight

No singularities !

Cartesian - Space

Desired pose $[{}^{S}_{T}T_0 \dots {}^{S}_{T}T_n] \rightarrow$ 1D trajectory gen.
→ inverse Kin at each execution instant
→ calculated joint values in live → sent to controller

- Motion to Cartesian pose with interpolation of Cartesian coordinates



⚠ Interpolating orientations require special care
→ Result must be orthonormal for rot matrices
→ Interpolating fixed angle and euler is same
→ Quaternion and angle axis can be interpolated
→ slerp function

Cartesian-space trajectories allow not only straight lines but also curves and splines

Issues with cartesian-space

1. Intermediate point not reachable
2. High joints speeds near singularities
3. Start and end have diff. arm config.
   → also in joint-space
4. Slower execution

---

5. Parallel robots

Definition and use

>= 2 kinetic chain linking fixed base to end effector
→ Actuators attached to base (generally)
→ Often passive joints
→ Links transmit forces along main axis

Advantage / disadvantage

Series: + large workspace
+ simple forward kin and singularity avoidance
- complex inv kin
- Payload-to-robot ratio < 0,15 → heavy links

Parallel: + simple inv. kin
+ High speed (low inertia), high precision
- reduced workspace
- Difficult singularities and forward kin

DOF

$$n = 6(b-j-1) + \sum_{i=1}^{j} f_i$$

- With
  - n the number of degrees of freedom of the mechanism
  - b the number of bodies (links) including the base
  - j the number of joints
  - f_i the number of degrees of freedom of joint i

Inverse Kin $\Theta = L^{-1}(X)$

→ simple geometric solutions

Forward Kin

→ derived from inverse kin
→ many solutions exists
(many cart. pose with same joint config)
→ most of time use numerical methods
→ sometimes close form exists like for Delta robot

Instant Kin

- Therefore it is easier to derive the inverse Jacobian matrix from the inverse kinematics

$$\boxed{\delta\Theta = {}^{B}J^{-1}(\Theta)\ \delta^{B}X} \qquad \delta\theta = \begin{bmatrix} \frac{\partial\theta_1}{\partial x} & \frac{\partial\theta_1}{\partial y} & \cdots \\ \frac{\partial\theta_2}{\partial x} & \frac{\partial\theta_2}{\partial y} & \cdots \\ & \vdots & \end{bmatrix} \delta X$$

- The Jacobian matrix can be found by inversion: $J(\Theta) = (J^{-1}(\Theta))^{-1}$

Singularities

Serial

Joint motion do not generate end-effector motion
$|J(\theta)| = 0$

Parallel

End-effector motion possible without joint move.
→ lost control, high forces

full jacobian matrix det is 0 $|J^{-1}(\theta)| = 0$
↳ (also includes passives joints)

6. End effector (Grippers, tools)

End effect. interface

⇒ Three characteristics

→ Strength

→ Compliance

→ Overload protection

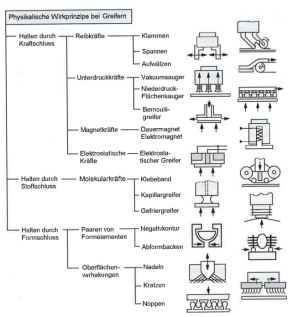
Selection

- Factors that should be consider in assessing gripping requirements
  - The part surface to be grasped must be reachable
  - The size variation of the part must be accounted for
  - The gripper design must accommodate the change in size that occurs between part loading and unloading
  - Consideration must be given to the potential problem of scratching and distorting the part during gripping, if the part is fragile or delicate surface
  - If there is a choice between 2 different dimensions on a part, the larger dimension should be selected for gripping
  - Gripper fingers can be designed to conform to the part shape by using resilient pads or self-aligning fingers

physical principle		impactive mechanical				pneumatic	magnetic	
prehended object	gripper type	parallel gripper	radial gripper	angle gripper	3 point gripper	suction gripper	permanent magnet	electro-magnet
mass	0.2 to 1 kg	■	■	■	■	■	■	
	1 to 10 kg	■	□	■	■	■		
	10 to 50 kg	◧	□	■	■	■		■
	heavier than 50 kg	□		■	■	◧		
dimensions	20 to 50 mm	■	■	■	■	■	■	■
	50 to 300 mm	■	□	■	◧	■	■	■
	300 mm to 1 m	■		■	□	■		■
	more than 1 m	■		■		■		■
inner grip surfaces		■		□	■			◧
surface	polished	■	■	■	■	■	■	■
	rough	■	■	■	■			◧
	porous	■	□	□	□	□		◧
	sensitive	□			□	■	■	■
round parts	disk	◧	■		■	■	■	■
	short cylinder	■	■	■	■	■		■
	shaft, rod	◧		■				◧
prismatic parts	block part	■	■	■		■	■	◧
	flat/short	□	■	□		■	■	■
	flat/long			□		■	◧	■
synthetics		■	□	□		■		
textiles						□		
foil						◧		
glass		□	◧	◧	◧	■		
stoneware		□	◧	◧	◧	□		
sheet metal		◧		◧		■	■	■

drive system / evaluation criteria	mechanical	pneumatic	hydraulic	magnetic	electric motor
high gripping force	◐	○	●	◐	○
controllability	○	○	◐	●	●
energy transmission	●	◐	○	●	●
insensitivity to dirt	●	◐	●	○	◐
maintanence	●	◐	◐	○	◐
emergency stop behaviour	●	◐	◐	○	○
constructional size	○	○	●	◐	◐
environmental influences	●	◐	○	●	●
costs	●	●	◐	◐	○

Gripping method	Non-penetrating	Penetrating
Impactive	Clamping jaws, chucks, collets	Pincers, pinch mechanisms
Ingressive	Brush elements, hooks, hook and loop (Velcro)	Needles, pins, hackles
Contigutive	Chemical adhesion (glues), surface tension forces	Thermal adhesion
Astrictive	Electrostatic adhesion	Magnetic grippers, vacuum suction



- Impactive: impact of jaws against object surfaces (mechanical)
- Ingressive: surface deformation or even penetration (intrusive)
- Contigutive: direct contact (e.g. chemical, thermal adhesion)
- Astrictive: binding forces (e.g. vacuum, magnetic, electrostatic)

7. Vision

What is a digital image ?

Pixel coordinate: (u, v) = (column, row)
→ origin at top left corner

> An image is a matrix of number

Matrix coordinates: (r, c) = (row, column)

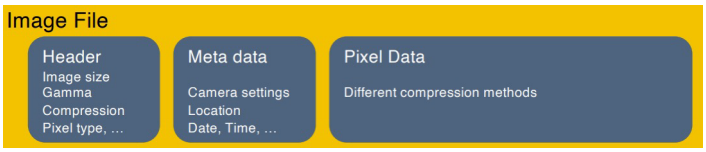


Image compression

Lossless : identify patterns and store params

Lossy : exploits human vision limitations
→ only one way.

I. Image processing

Input images → output more useful image

One-to-one operations

Monadic

apply a function for each pixel separatly * parallel processing possible

Functions → change data type [0, 255] → [0.0, 1.0]
brightness $f(x) = x + 0.5$, Δ saturation
contrast $f(x) = 2x$, Δ saturation
negative $f(x) = 1 - x$
posterization (quantization) $f(x) = \frac{floor(x \cdot N)}{N}$
histogram normalization (non linear mapping)
gamma correction $f(x) = x^{\gamma}$ highlight image details
in matlab image .^ gamma
thresholding → boolean operation

Diadic

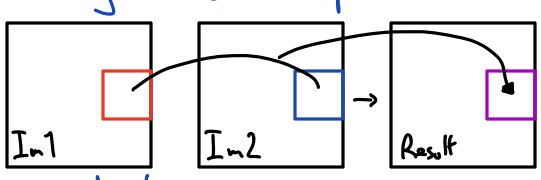
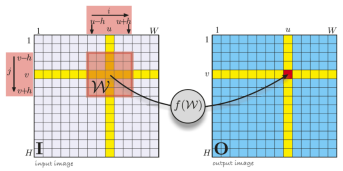


image subtraction
green screen operations (masking)

Spatial operations



Use a Kernel (window) to define the output pixel
Size : 2h+1, odd

Isotropy : in all directions the same

Handling edge pixels : either copy paste or delete

Applying the kernel :

Correlation

$$O[u,v] = \sum_{(i,j) \in W} I[u+i, v+j] K[i,j], \forall (u,v) \in I \quad i \in [-h \dots h], j \in [-h \dots h]$$

Convolution

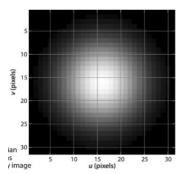
$$O[u,v] = \sum_{(i,j) \in W} I[u-i, v-j] K[i,j], \forall (u,v) \in I \quad i \in [-h \dots h], j \in [-h \dots h]$$

$O = K \otimes I$

Scale factor : $S = \Sigma |K_{i,j}|$
We typically want a scale of 1 ! $K = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

Gaussian kernels

$K : G(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$

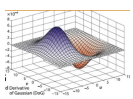


Act as smoothing, blurring
σ control width

Derivative (Finding edges)

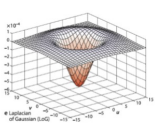
Perform Gaussian before derivative (sensitive to noise)
→ Use kernel that perform derivative
→ Sobel kernel $K = \frac{1}{2}(-1\ 0\ 1) \Rightarrow \frac{I(u+1) - I(u-1)}{2}$

DoG Derivative after gaussian



Laplacian isotropic second derivative

LOG Laplacian after gaussian



Template matching

→ detect if kernel is same as image and give a score

Filtering

Remove speckle noise or salt and pepper noise
→ Replace pixel by a mean or median of its kernel

Mathematical morphologies

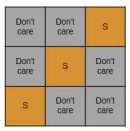
Non-linear spatial operator
Each pixel is a function of a subset pixels of kernel
Output image contains shapes that are compatible with the structuring element S
S can have various shapes (rectangle, or matching shape)

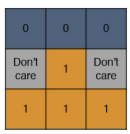
Erosion
only shapes that contain S survives
large objects become smaller

Dilation
any pixel under S is true, output is true
make object larger

Opening
Erosion followed by dilation
Clean up shapes

Closing
Dilation → Erosion
Fill holes and connect objects

Avanced mathematical morph.




- → Detect diagonal line
- → Circle structuring elements → round corners
- → Hit or miss pixel window must fit 0 and 1's →
- → Skeletonization iterative morph operations

II Feature extraction - regions
Group of connected pixels, and same color (binary)

Image moments (binary image)
Computationally cheap class of images features

$$m_{pq} = \sum_{(u,v)\in I} u^p v^q I[u,v]$$

coordinate; pixel value, $[0;1]$

Zeroth / first moments
Area ↓ ; weighted average ↘

$$m_{00} = \sum_{(u,v)} I[u,v] \qquad m_{10} = \sum_{(u,v)} u I[u,v] \qquad m_{01} = \sum_{(u,v)} v I[u,v]$$

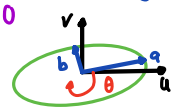
Center of mass
$$u_c = \frac{m_{10}}{m_{00}} \qquad v_c = \frac{m_{01}}{m_{00}}$$

Blobs
region of connected components (same color)
In general a blob is assigned a label, i.e each pixel of that blob has same label.

Object features
$A_{N\times N}$ $v_{N\times 1}$ $\begin{cases} v' = Av \\ v' = \lambda v \end{cases}$

Eigenvector: vectors whose length are scaled by λ from matrix A but direction is unchanged

$$\det(A-\lambda I) = 0 \qquad (A-\lambda I)v = 0$$

Equivalent ellipse method


Find orientation and shape of blob
Idea: fit an ellipse with same centroid and moment of inertia about centroid

$$J = \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix} \qquad \mu_{pq} = \sum_{(u,v)\in I} (u-u_c)^p (v-v_c)^q I[u,v]$$

(centroid)

Ellipse axis ← eigenvalues of J

$$a, b = 2\cdot\sqrt{\frac{\lambda_{1,2}}{m_{00}}}, \quad \lambda_1 > \lambda_2$$

Orientation ← eigenvector of λ_1

$$\theta = \tan^{-1}\left(\frac{v_x}{v_y}\right)$$

Perimeter and circularity
Compute perimeter P → circularity $\rho = \frac{4\pi\mu_{00}}{P^2}$

Hu moments
$\gamma = \frac{1}{2}(p+q)+1$

7 moments invariant from scale, rotation, translation
→ based on normal moments but normalized with

II Feature extraction - lines
Lines lay along obvious boundaries in the scene
1. Convert to grayscale 2. Detect image edges 3. Hough
(sobel kernel / canny)

Canny edge detector
1. Gaussian 2. Gradient filter 3. Find ridges by keeping the pixel with max gradient in that direction
4. Apply two thresholds to detect strong and weak edges
5. Suppress weak edges not connected to strong edges

Hough line transform
Using polar line representation $u\cdot\sin(\theta) + v\cdot\sin(\theta) = \rho$
Number of potential lines are spanned through the image
Each line get a point when touching a pixel
→ Lines with best scores are special in the image

II Feature extraction - points
Detect points of interest such as corners etc
Application: camera calibration, tracking ..

Moravec's interest operator
Idea: corners are areas with significant variation in intensity in all directions
→ kernel in several directions to calculate intensity changes

Harris corner detector
Compute gradient filter and based on weighted gaussian matrix of the pixel gradient tensor
Use the det and trace of the matrix to define eigenvalues and detect corners based on λ_1, λ_2 when λ_1 and $\lambda_2 \gg$ threshold

Feature matching
Extract a window centered on a corner point
Use similarity measure to detect the same window in another image
→ Real time tracking

SIFT-Features Scale Invariant Feature Transform

1. Scale-space extrema detection
   Use multiple image size (octave) with different gaussian blur and apply gaussian filter (DoG) in "3d"
2. Keypoint localization
   Use same principle as Harris to detect corners
3. Orientation assignment
   Compute image gradients near keypoint and find orient.
4. Keypoint descriptor
   Orientation, gradient magnitudes, main gradient...

Applications Object detection, panorama stitching

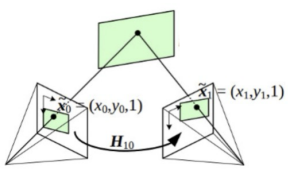
III Further algorithms

Image scaling
Select each n^{th} pixel in each column, row

Aliasing problem
When downsampling image, high freq will appear in the low frequencies → artefacts like jagged edges or moiré
Solution: gaussian blur before subsampling

Image warping
Change pixel coordinates $u' = f_u(u,v)$ $v' = f_v(u,v)$
→ sometimes interpolation necessary $u = 213.33$ ✗

Perspective transform
Use a homography matrix, relates the transformation between two planes seen from different view



IV Visual servoing
Control pose of robot's end effector relative to goal using visual features extracted from image
→ need 2 cameras for 3 DOF tasks
→ compensate errors in model and sensors
→ need 3 points to define motion

Optical flow Detect pixel velocity
However, 2 unknown cases:
Perceptibility: Zero change for some points → motion not perceptible
Ambiguity: Distinct camera movements cause same pixel behaviour → add additional sensors

8. Robot dynamics

Dynamics
$^A p = m\,^A v_c$ $^A F = m\,^A \dot{v}_c$
→ rate of change in momentum proportional to force
$^A L = {}^A I\,{}^A w$ $^A N = {}^A \dot{I} \cdot w + {}^A w \times {}^A I\,{}^A w$
→ rate of change in ang. momentum prop to torque N
Goal: get joint torque in motion

Methods to derive dynamics

Iterative Newton-Euler method
① Outward iteration: calculate all bodies velocities and accelerations beginning from B
Initial cond. ${}^0w_0 = 0$ ${}^0\dot{w}_0 = 0$ ${}^0\dot{v}_0 = G$

② Inward iteration: calculate all forces and torque required to move bodies with these velocities and accel. $\begin{bmatrix} {}^{n+1}f_{n+1} \\ {}^{n+1}n_{n+1} \end{bmatrix} = \mathcal{F}$
→ Popular for serial robots

Lagrangian dynamic formulation
Calculate kinetic and potential energies of the entire robot → partial derivatives of these energies lead to joint torque.
→ only consider dominant bodies
→ must be used for parallel robots

State-space equations
Result of the iterations can be described by:

$$\mathcal{I} = M(\Theta)\ddot{\Theta} + V(\Theta,\dot{\Theta}) + G(\Theta) + J^T(\Theta)\mathcal{F}$$

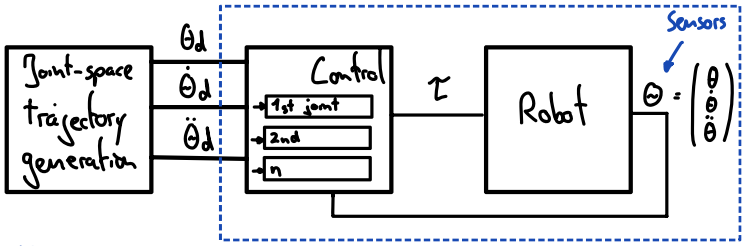
$\mathcal{I}$: Vector of joint torques
$M(\Theta)\ddot{\Theta}$: mass matrix $n \times n$
$V(\Theta,\dot{\Theta})$: vector of centrifugal and Coriolis terms $n \times 1$
$G(\Theta)$: Vectory of gravity terms $n \times 1$
$J^T(\Theta)\mathcal{F}$: vector of TCP forces/torques $n \times 1$

Model not including bending and friction

Cartesian space
$$\mathcal{F} = M_X(\Theta)\ddot{X} + V_X(\Theta,\dot{\Theta}) + G_X(\Theta)$$
$\ddot{X} = J\ddot{\Theta} + \dot{J}\dot{\Theta}$ V_X and G_X, calculated from V, G as cartesian

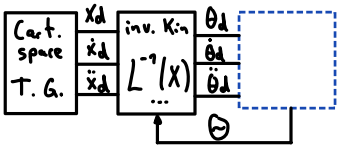
Parallel robot dynamics
$$\mathcal{I} = M(\Theta)\ddot{\Theta} + J^T(\Theta)\left[M_X(X)\ddot{X} + V_X(X,\dot{X}) + G_X(X)\right]$$

9. Control



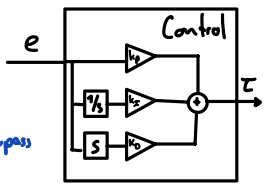
High performance always need feedback control
MIMO system → need to be simplified to multiple SISO

Linear Control



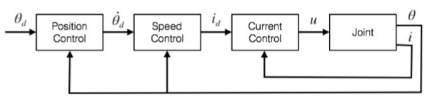
Cartesian space control requires inv. Kin and

$\dot{\theta}_d = J^{-1}\dot{x}_d \quad \ddot{\theta}_d = J^{-1}\ddot{x}_d - J^{-1}\dot{J}\dot{x}_d$



PID $\tau = k_p e + k_D \dot{e} + K_I \int e\,dt$
↳ ∂ noise → lowpass

Cascading
→ better tuning



Model-Based
→ with linear SISO model

Non-Linear Control "Computed torque"

→ Better compensate inertia, damping

Model-based
Design a controller for $\ddot{X}$ or $\ddot{\Theta}$ the acceleration only

$\tau = M(\Theta)\ddot{\Theta} + V(\Theta,\dot{\Theta}) + G(\Theta) + J^T(\Theta)\mathcal{F}$

$\tau' = \alpha\ddot{\Theta} + \beta$ ("decoupling" → feed forward ⇒ often neglected)

$\ddot{\Theta} = \ddot{\Theta}_d + K_v(\dot{\Theta}_d - \dot{\Theta}) + K_p(\Theta_d - \Theta) + K_I\int E\,dt$

($\ddot{E}$, $\dot{E}$, E)

Feed-forward + Gain scheduling
↳ β is a feed-forward → live adaptation can improve perf. → adaptive control
↳ Schedule different gains based on operating point

Cartesian Control

Do full control in cartesian space → less intuitive
Uses J^{-1} or J^T and $\mathcal{F}_e$ or cartesian dyn. model

Force control → need torque sensor

→ sometimes compliance in gripper is sufficient
Hybrid position-force control: when touching an object → the force control is activated
Impedance control: stiffer in some directions

10. Application analysis

Use robots for dangerous, dirty, dreary tasks DDD

1. Kinematics required; DOF, workspace, accuracy
2. End-effector; payload
3. Fixtures and external sensors, vision
4. Path-planning and control; sketch trajectories singularities; force control
5. Costs