Robotic1 Theory

domenica 9 gennaio 2022

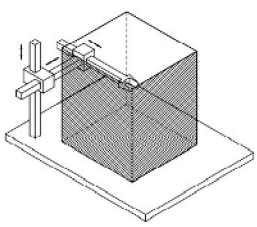
Classification of robots by mechanical structure: { Robot maipulator with fixed base → / Mobile robot with a mobile base ↓

There are two categories of robots:

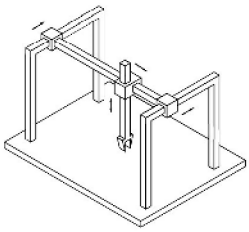
- **Mobile robots** with wheels, consist of a rigid body and the wheels allow movement on the ground;
- **Legged mobile** robots whose ends are not always in contact with the ground.

The wheels used by the first type can be:

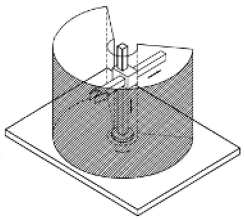
- **Fixed** wheels with a single axis of rotation which coincides with the center of the wheel;
- **Steering** wheel allows you to rotate vertically around the center of the wheel;
- **Caster** wheel has its vertical axis shifted by a constant offset;
- **Mecanum** (swedish) wheel is a fixed wheel with rollers on the edge placed at 45 ° so as to be a mobile robot with 4 of these being omnidirectional.



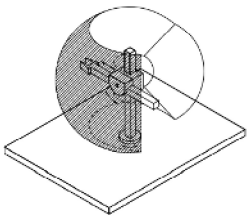
1. Cartesian manipulator



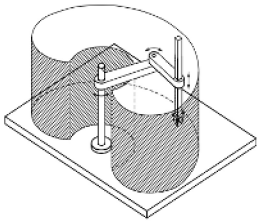
2. Gantry manipulator



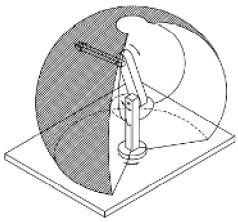
3. Cylindrical manipulator



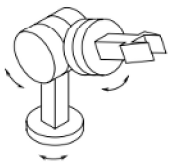
4. Spherical Manipulator



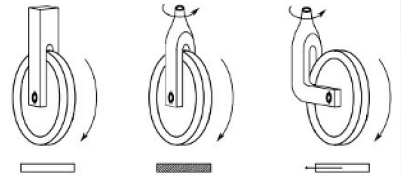
5. SCARA



6. Anthropomorphic manipulator



7. Spherical wrist



Left to right: Fixed wheel, Steering wheel, Caster wheel and mecanum wheel

Are formed by a sequence of rigid bodies (links) interconnected by joints (joints), thus forming arms that allow movemenat while the wrist allows dexterity and the end-effector performs the task. If there is only a sequence of links joining the two ends of the chain it is called an open kinematic chain otherwise it is a closed kinematic chain. The joints can be of two types: prismatic and revolutionary, each joint adds a degree of freedom (DOF) except in cases of closed kinematic chains. To position or orient an object in 3d space you need to have 3 DOFs while to position and orient an object you need 6 DOFs.
The manipulators are divided according to the topological structure of the joints starting from the base and are:

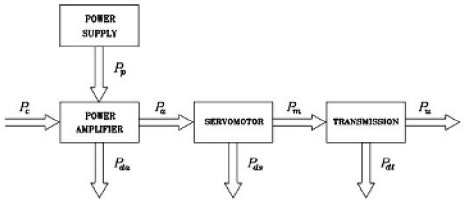
1. **Cartesian** manipulator (**P-P-P**) offers high accuracy, good mechanical rigidity but low dexterity;
2. **Gantry** manipulator (**P-P-P-P**) similar to the previous one but has a top approach;
3. **Cylindrical** manipulator (**R-P-P**) good mechanical rigidity but accuracy decreases horizontally;
4. **Spherical** manipulator (**R-R-P**) low mechanical strength and accuracy decreases radially;
5. **SCARA** (Selective Compliant Arm for Robotic Assembly) (**R-R-P**) all axes are parallel unlike the first high strength at vertical loads while it is weak at horizontal loads therefore perfect for assembly;
6. **Anthropomorphic** manipulator (**R-R-R**) the last two axes are parallel there is no relationship between DOFs and Cartesian variables and accuracy varies in the workspace;
7. **Spherical wrist** (**R-R-R**) the axes are orthogonal and intersect at a point this allows to divide the task in positioning and orienting

Industrial robots are robot manipulator with a limited autonomia in a applied in a structured environment are used in 3 levels of automation:

- Rigid automation, high production of the same type as it does not use a fixed operation sequence;
- Programmable automation, production of small-medium batches of different types as it is easy to change the sequence of operations;
- Flexible automation is the evolution of the previous one, the goal is to produce several batches of different products trying to minimize the reprogramming time of the operational sequence.

Advanced robotics refers to robots with particular characteristics of autonomy, which operate in structured or non-structured environments, whose physical and geometric characteristics are not known a priori. Advanced robots are used in places where human operators would not be safe or unavailable.Field robots are used in contexts where humans may not survive or be subjected to unbearable risks such as exploring volcanoes. Service robots are used in civilian contexts such as cohabitation with humans or as a tour guide or as a mass transport of people. Modeling a robot manipulator is necessary to find a motion control strategy. The kinematic analysis describes the analytical relationship between the position of a joint and the effector-end while the differential kinematics determines the analysis of the movement in terms of speed. Kinematics allows to study two problems: direct and inverse kinematics.

The motion imposed to a manipulator's joint is realized by an actuating system which in general consists of: { power supply, / power amplifier, / servomotor, / transmission



Components of a joint actuating system

$\dot{q}_{ll} = \frac{1}{n}\dot{q}_{lm}$; $\tau_{ll} = n\tau_{lm}$ Relationship betwen reduction caused by the trasmission

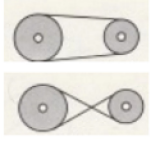
TRASMISSION

The execution of **joint motions** of a manipulator **demands** low speeds with high torques. In general, such requirements do not allow an effective use of the mechanical features of **servomotors**, which typically **provide** high speeds with low torques in optimal operating conditions. It **is** then **necessary** to interpose a **transmission** (gear) to optimize the transfer of mechanical power from the motor (**Pm**) to the joint (**Pu**). During this transfer, the power **Pdt** is dissipated as a result of friction.
The typically trasmissions used for industrial robots:

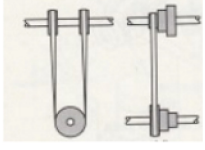
1) **Spur gears** that modify the characteristics of the rotational motion of the motor by changing the axis of rotation and/or by translating the application point; spur gears are usually constructed with wide cross-section teeth and squat shafts. (used in revolute and prismatic joints)
2) **Lead screws** and **worm gearing** that convert rotational motion of the motor into translational motion. (used in prismatic joint)
3) **Timing belts** and **chains** useful for dislocate the motor from the joint, belts is better use for high velocity purpose instead the chains cause more vibration (more weight) for high torqes use the chain cause the belts cause strain by the stress.
4) **harmonic drive** can generate high torques thanks to an high ratio reduction (up to 320:1) and without any backlash.

1) Spur gears

2) Lead screws (left) and Worm gearing (right)



3) Timing belts and chains



4) Harmonic drive

To **get** a final **acceleration** "*a*", the **engine** must **exert** an acceleration equal to: "***n*×*a***". therefore the motor must make a torsional force equal to: $T_m = J_m \ddot{\theta}_m + 1/n\,(J_u \ddot{\theta}_u) = (J_m n + J_u/n)\,a$

We find the minimum value of n then we derive the function according to n: $\frac{\partial T_m}{\partial n} = (J_m - J_u/n^2)\,a = 0$ So the best matching condition with the inertias is: $n = (Ju + Jm)^{1/2}$

SERVOMOTORS

Actuation of joint motions is entrusted to motors which allow the realization of a desired motion for the mechanical system. Concerning the kind of input power Pa, motors can be classified into three groups:

1. **Pneumatic** motors are **difficult to control accurately** so used for gripper or artificial muscle.
2. **Hydraulic** motors, advantages: **no static overheating**, **self-lubricated**, **inherently** safe, **excellent power-to-weight ratio**, **large torques at low velocity** (w/o reduction); disadvantages: **needs hydraulic supply**, **large size**, **linear motion only**, low power conversion efficiency.
3. **Electric** motors, are made with a stator, a rotor and a commutator (built with brush or not). Advantages: **power supply available**, **low cost**, **large variety**; disadvantages: **overheating in static conditions**, need special protection in

N.B. Requirement of all the type of motor in conventional applications:

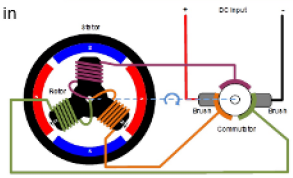
- Low inertia and high power-to-weight ratio,
- Possibility of overload and delivery of impulse torques,
- Capability to develop high accelerations,
- Wide velocity range (from 1 to 1000 RPM ≃ 210 rad/s),
- High positioning accuracy (at least 10^{-3} of a round ≃ 6×10^{-3} rad),
- Low torque ripple, to guarantee continuous rotation at low speed.

2. **Hydraulic** motors, advantages: **no static overheating, self-lubricated, inherently** safe, **excellent power to weight ratio, large torques at low velocity** (w/o reduction); disadvantages: **needs hydraulic supply, large size, linear motion only**, low power conversion efficiency.
3. **Electric** motors, are made with a stator, a rotor and a commutator (built with brush or not). Advantages: **power supply available, low cost, large variety**; disadvantages: **overheating in static conditions**, need special protection in flammable environments. Electric motors may require direct current or alternating current as their power supply:

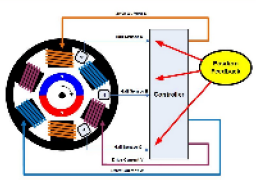
a. DC:
- **High startup power and torque,**
- **Fast response times** to starting, stopping and acceleration,
- **Availability** in several standard **voltages,**

b. AC:
- **Low startup power** demands that also protect components on the receiving end,
- **Controllable starting current** levels **and acceleration,**
- Capabilities for **multi-phase configurations,**
- **High durability** and longer life spans,
- **Velocity Frequence Drive control speed and torque at different stages of use,**

- Wide velocity range (from 1 to 1000 RPM ≃ 210 rad/s),
- High positioning accuracy (at least 10^{-3} of a round ≃ 6×10^{-3} rad),
- Low torque ripple, to guarantee continuous rotation at low speed.



Brushed DC motor



Brushless DC motor

Brushless advantages over brushed:
- reduced losses (electrical and mechanical),
- more compact rotor,
- reduced maintenance (no substitution of brushes),
- easier heat dissipation,

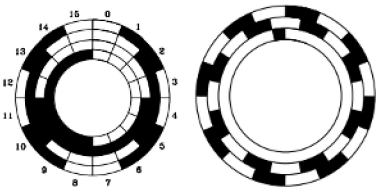
SENSORS

Properties of measurement systems: **accuracy** (improved with **calibration**), **repeatability** (depends on robot controller/measurement **resolution**), **stability**, **linearity** error, **offset** error and **resolution error**.

The sensors could be classify in:

1. **Proprioceptive** sensors measure the internal state of the robot (position and velocity of joints, torque at joints and acceleration of links)

- **Position** sensors:
  a. **Absolute encoder** rotating optical disk, with alternate transparent and opaque sectors on $\mathbf{N_t}$ (~13) concentric **tracks** in **grey code**, light beams are emitted by leds and sensed by photo-receivers, **resolution = 360° / 2^{Nt}**, **ready to measure at start** (no homing),
  b. **Incremental encoder** optical rotating disk with three tracks, alternating transparent and opaque areas, **Ne** (~100÷5000) **pulses** per turn, to improve resolution (×4) the leading and trailing edges of signals A and B are used, **resolution = Ne×4**, A and B tracks (channels) are in quadrature allowing to detect the direction of rotation, Z track needed to reset and start the counter (needs "homing" at start),

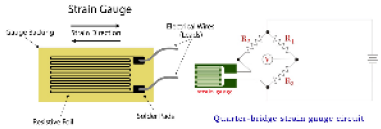


Absolute encoder (left) and Incremental encoder (right)

- **Velocity** measures:
  a. **Indirect measure with numerical differentiation** of digital measures of position (Backward Differentiation Formulas/Kinematic Kalman Filter),
  b. **Tachometer** mounted on the (electrical) motor axis, measures the derivative of the magnetic flux (B = cos(ωt) → **dB/dt** = **V** = B×**ω**×sin(ωt)) that passes through the coil during rotation because it is proportional to the angular velocity (V α ω),
- **Accelerometers** sensors:
  a. use different principles for converting mechanical motion in an electrical signal:
     - **piezoelectric**: piezoceramics (PZT) or crystals (quartz), better linearity & stability, wide dynamic range up to high frequencies, no moving parts, no power needed,
     - **piezoresistive**: for high-shocks, measures also static acceleration (g0), needs supply,
     - **capacitive**: silicon micro-machined sensing element, superior in static to low frequency range, can be operated in servo mode, cheap but limited resolution,
  b. **MEMS** (Micro Electro-Mechanical Systems).

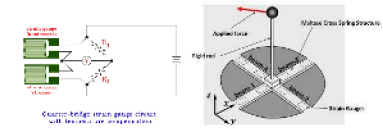
2. **Exteroceptive** sensors measure/characterize robot interaction with the environment, enhancing its autonomy (forces/torques, proximity, vision)

- Force sensors:
  a. **Strain gauges** are sensitive to small stresses occurring in the object by measuring the change in resistance. They are also used in **Wheatstone configuration** for **a direct output** of the **stressed force**, to **avoid variations** in **resistance** with **temperature** it is possible to **replace** the resistance **R2** with another **strain gauge** in a stressless place. By making a **Maltese-cross configuration** (two pairs of strain gauges are mounted on opposite sides of the 4 deformation elements) it is possible to **calculate the force and the torques** along **x-y-z**.
  b. **RCC** (Remote Center of Compliance) placed on the wrist so as to introduce passive "compliance" to the robot end-effector, in response to static forces and moments applied from the environment at the contact area.



Structure of Strain gauge (left) and Strain gauge in Wheatstone configuration (right)

- Proximity sensors:
  a. **Sonar** uses acoustic pulses and their echoes to measure the range of an object (d_O), since the speed of sound (c_s) is usually known for a certain media (air, water), the range of an object is proportional to the travel time (t_v) of the echo, $d_O = c_s \times t_v/2$
  b. There are two types of commonly used **laser-based range sensors**: **time-of-flight sensors and triangulation sensors**. Similar to sonars, time-of-flight sensors calculate distance by measuring the time it takes for a pulse of light to travel from the source to the observed target and then to the detector (usually placed with the source). The travel time multiplied by the speed of light (correctly adjusted for the air temperature) gives the measure of the distance. In the case of triangulation sensors, the laser beam emitted by a photodiode is projected onto the observed surface and the reflected beam is focused on a CCD sensor by means of a special lens. The position of the focused beam reflected at the receiver gives rise to a signal proportional to the distance of the transmitter from the object. In fact, from the measurement of the CCD sensor it is possible to use the angle at which the reflected energy hits the sensor. Once the relative position and orientation of the CCD sensor with respect to the photodiode is known (through calibration) it is possible to calculate the distance from the object.

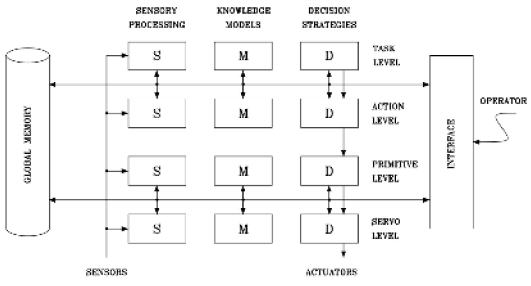


Wheatstone with double strain gauge (left) and Maltese-cross configuration (right)

- Vision sensors:
  a. CCD b. CMOS c. Camera (a more complex system of chip)

SUPERVISION AND CONTROL ARCHITECTURES

1. The first generation has been characterized by programming techniques of **teaching-by-showing** type. The operator guides the manipulator manually or by means of a teach pendant along the desired motion path. During motion execution, the data read by joint position sensors are stored and thus they can be utilized later as references for the joint drive servos; in this way, the mechanical structure is capable of executing (playing back) the motion taught by a direct acquisition on the spot,
2. The need for interaction of the environment with physical reality has imposed integration of several functions, typical of high-level programming languages (BASIC, PASCAL), with those specifically required by robotic applications. Many **robot-oriented languages** have kept the teaching-by-showing programming mode. Since the general framework is that of a computer programming environment, two alternatives have been considered:
   - to develop ad hoc languages for robotic applications,
   - to develop robot program libraries to supporting standard programming languages.
3. The last programming environment allows access at the task level to a functional architecture reference model characterized by an object-oriented language. Such an environment should have the ability to specify a task by means of high-level instructions that allow the automatic execution of a certain number of actions on the objects present in the scene. This generation of programming languages for robots is under development, belongs to the field of expert systems and artificial intelligence.



Reference model for a control system functional architecture

ROTATION MATRIX

$$R = [\,x'\ \ y'\ \ z'\,] = \begin{bmatrix} x'_x & y'_x & z'_x \\ x'_y & y'_y & z'_y \\ x'_z & y'_z & z'_z \end{bmatrix} = \begin{bmatrix} x'^{T}x & y'^{T}x & z'^{T}x \\ x'^{T}y & y'^{T}y & z'^{T}y \\ x'^{T}z & y'^{T}z & z'^{T}z \end{bmatrix} \quad (2.3), \qquad R^{T}R = I_3 \quad (2.4), \qquad R^{T} = R^{-1} \quad (2.5), \qquad \det(R) = 1,$$

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.6), \qquad R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\gamma\beta \\ 0 & 1 & 0 \\ -\sin\gamma\beta & 0 & \cos\beta \end{bmatrix} \quad (2.7),$$

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\gamma & \cos\gamma \end{bmatrix} \quad (2.8),$$

The column vectors of R are the direction cosines of the axes of the rotated frame (x'y'z') with respect to the original frame (xyz)

a.

A rotation matrix has three equivalent geometrical meanings: (the characteristic of having a common origin is implied below)

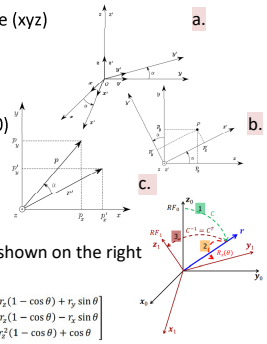
$R_x(\gamma) = \begin{bmatrix} 0 & c\gamma & s\gamma \\ 0 & -s\gamma & c\gamma \end{bmatrix}$ (2.8), The column vectors of R are the direction cosines of the axes of the rotated frame (x'y'z') with respect to the original frame (xyz)

A rotation matrix has three equivalent geometrical meanings: (the characteristic of having a common origin is implied below)

a. It describes the mutual orientation between two coordinate frames.

b. It represents the coordinate transformation between the coordinates of a point expressed in two different frames.

c. It is the operator that allows the rotation α of a vector (p') in the same coordinate frame.



${}^0R_1$

$p = R_z(\alpha)p'$ (2.10)

$p = R_z(\alpha)p'$

${}^0p = ({}^0R_1\,{}^1R_2\,{}^2R_3)\,{}^3p = {}^0R_3\,{}^3p$ --> 63 products + 42 summations; ${}^0p = {}^0R_1\,({}^1R_2\,({}^2R_3\,{}^3p)) = {}^0R_3\,{}^3p$ --> 27 products + 18 summations

AXIS/ANGLE REPRESENTATION

${}^0p = R(\theta, r)\,{}^1p$ RF_1 is the result of rotating RF_0 by an angle θ around the unit vector r, so we have a rotation elementary and two rotation to align z with r, shown on the right

Direct problem

$R(\theta, r) = C\,R_z(\theta)\,C^T$ the shape of C = [n s r] such as n×s = r so $R(\theta, r) = [n\ s\ r]\begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} n^T \\ s^T \\ r^T \end{bmatrix} = nn^T + ss^T \quad sn^T - ns^T = rr^T + (I - rr^T)c\theta + S(r)s\theta =$

$\begin{bmatrix} r_x^2(1-\cos\theta)+\cos\theta & r_xr_y(1-\cos\theta)-r_z\sin\theta & r_xr_z(1-\cos\theta)+r_y\sin\theta \\ r_xr_y(1-\cos\theta)+r_z\sin\theta & r_y^2(1-\cos\theta)+\cos\theta & r_yr_z(1-\cos\theta)-r_x\sin\theta \\ r_xr_z(1-\cos\theta)-r_y\sin\theta & r_yr_z(1-\cos\theta)+r_x\sin\theta & r_z^2(1-\cos\theta)+\cos\theta \end{bmatrix}$

Inverse problem

$\theta = \text{atan2}(\pm\sqrt{(R_{12}-R_{21})^2 + (R_{13}-R_{31})^2 + (R_{23}-R_{32})^2}, R_{11}+R_{22}+R_{33}-1)$ or $\theta = \text{acos}((R_{11}+R_{22}+R_{33}-1)/2)$ $r = 1/(2\,s\theta)\begin{bmatrix} R32 - R23 \\ R13 - R31 \\ R21 - R12 \end{bmatrix}$

EULER ANGLES (moved axes)

It defines a rotation of 3 angles (αβγ) about the axes (ZX'Z") in the moved Reference Frame (=at the current moved axes), so the obtained rotation matrix is: R = Rz(α) Rx(β) Rz(γ)

ROLL-PITCH-YAW ANGLES (fixed axes)

It defines a rotation of 3 angles (αβγ) about the axes (XYZ) in the fixed Reference Frame (=at the original axes), so the obtained rotation matrix is: R = Rz(α) Ry(β) Rx(γ) **Notice the order of the multiplication matrix is inverted**

HOMOGENEOUS TRANSFORMATION

It defines both the position and the orientation of one frame with respect to another, so it is a roto-translational operator: $T = \begin{bmatrix} R_1^0 & o_1^0 \\ 0^T & 1 \end{bmatrix}$ (2.42), where o_1^0 is the vector which describe the origin of Frame 1 with respect to Frame 0.

Instead to compute the inverse of T: $T^{-1} = \begin{bmatrix} R_0^1 & -R_0^1 o_1^0 \\ 0^T & 1 \end{bmatrix}$

DIRECT KINEMATICS

Choice of parameterization ***q***:

- ***n*** = # Degrees Of Freedom (DOF) = # robot joints
- **Unambiguous and minimal characterization of robot configuration** (angles or length of the joints)

Choice of parameterization *r*:

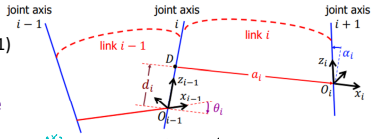
- Compact description of position and/or orientation (pose) variables of interest to the required task
- Usually, $m \le n$ and $m = 6$

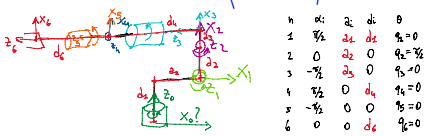
The structure of the direct kinematic function depends on the *r* choice. $r = f_r(q)$, if $f_r(q)$ is computed by:

❖ a geometric inspection in the general case with through the Denavit-Hartenberg (DH) table ❖ systematic if it multiplies all the homogeneous matrices of the joints

Definition of DH parameters

1. unit vector z_i along axis of joint i + 1,
2. unit vector x_i along the common normal to joint i and i + 1 axes (i ➔ i + 1)
3. α_i = twist angle from z_{i-1} to z_i around x_i, + if CCW, always constant
4. a_i = distance DO_i, + if oriented as x_i, always constant (= 'length' of link i)
5. d_i = distance $O_{i-1}D$, + if oriented as z_{i-1}, variable if joint i is **prismatic**
6. θ_i = angle from x_{i-1} to x_i around z_{i-1}, + if CCW, variable if joint i is **revolute**





h	α_i	a_i	d_i	θ
1	$\pi/2$	d_4	d_5	$q_1 = 0$
2	0	d_2	0	$q_2 = \pi/2$
3	$-\pi/2$	d_3	0	$q_3 = 0$
4	$\pi/2$	0	d_4	$q_4 = 0$
5	$-\pi/2$	0	0	$q_5 = 0$
6	0	0	d_6	$q_6 = 0$

INVERSE KINEMATIC

Solvability and robot workspace for tasks related to a desired end-effector Cartesian pose

1. Primary workspace **WS_1**: set of all positions *p* that can be reached with at least one orientation (ϕ or R):
   - out of WS_1 there is no solution to the problem,
   - if $p \in WS_1$, there is a suitable ϕ (or R) for which a solution exists.
2. Secondary (or dexterous) workspace **WS_2**: set of positions *p* that can be reached with any orientation (among those feasible for the robot direct kinematics):
   - if $p \in WS_2$, there exists a solution for any feasible ϕ (or R).

$WS_2 \subseteq WS_1$

Multiplicity of solutions (general case):

If $m = n$

A. ∄ solutions

B. A finite number of solutions (regular/generic case)

C. "Degenerate" solutions: infinite or finite set, but anyway different in number from the generic case (singularity)

If $m < n$ (robot is kinematically redundant for the task)

A. ∄ solutions

B. ∞^{m-n} solutions (regular/generic case)

C. A finite or infinite number of singular solutions

Possible methods to look for a solution:

ANALYTICAL solution (in closed form)

- Preferred, if it can be found
- Use ad-hoc **geometric inspection**
- **Algebraic methods** (solution of polynomial equations)
- Systematic ways for generating a reduced set of equations to be solved

NUMERICAL solution (in iterative form)

- certainly **needed** if $n > m$ (**redundant case**) or at/**close to singularities**
- **slower**, but **easier** to be set up
- in its basic form, it uses the (analytical) **Jacobian matrix** of the direct kinematics map
- Newton method, Gradient method, and so on

Newton method doesn't always **find a solution**, **when**: $J_r(q)$ **is not full rank** (singularity and not invertible) AND the error $e \in N(J_r^T(q))$ (e = pd-fr(q), i.e. $Jr^T(q)$ (pd-fr(q)) = 0) because isn't possible to update to the new step.

All the **numerical** solutions **generate only one solution**, therefore **for further solutions use different starting points**.

The **Newton** method **converges** much **faster** than the **Gradient** method but on the other hand the second method **always converges**, so the best algorithm would implement the Gradient for the initial steps and then switch to the Newton method.

If a manipulator has a **limit** on the **joints** they must be considered at the end.

$J_r(q) = \frac{\partial f_r(q)}{\partial q}$

Relationship between the Jacobian matrix and the function of the position of the E-E

$q^{k+1} = q^k + J_r^{-1}(q^k)\,[r_d - f_r(q^k)]$

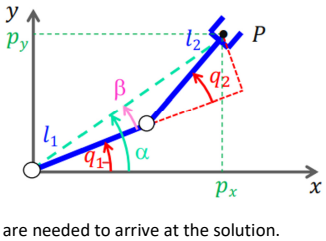
Iterative calculation of the position of the E-E with the newton method

$q^{k+1} = q^k + \alpha\, J_r^T(q^k)(r_d - f_r(q^k))$

Iterative calculation of the position of the E-E with the Gradient method

★ Example of inverse kinematics exercise on a 2R planar manipulator

Problem data: p = [0.7 0.7]', $l_1 = 1$, $l_2 = 1$ Unkwon q = ?, multiple solutions?



- Solution in **analytic** way:

  $p_x = l_1c_1 + l_2c_{12}$, $py = l_1s_1 + l_2s_{12}$ ➔ $p_x^2 + p_y^2 - (l_1^2 + l_2^2)$ ➔ $2l_1l_2(c_1c_{12} + s_1s_{12})$ ➔ $2l_1l_2$ ➔ $c2 = (p_x^2 + p_y^2 - (l_1^2 + l_2^2))/2l_1l_2 = -0.51 \in [-1,1]$

  $s_{2\pm} = \pm\text{sqrt}(1 - c_2^2) = \mp 0.86$ ➔ $q_{2\pm} = \text{atan2}(s_{2\pm}, c_2) = \mp 0.67$ ➔ $q_{1\pm} = \alpha \pm \beta$ ➔ $q_{1\pm} = \text{atan2}(p_y, p_x) \pm \text{atan2}(l_2s_2, l_1 + l_2c_2) = 1.30, -0.27$

- Solution in **numerical** way:

  $q^0 = [\pi/4, 0]$, $f_r(q) = [l_1c_1 + l_2c_{12};\ l_1s_1 + l_2s_{12}]$, $J_r(q) = [-(l_1s_1 + l_2s_{12})\ -l_2s_{12};\ l_1c_1 + l_2c_{12}\ \ l_2c_{12}]$ ➔ $e = pd - f_r(q^0) = [-0.71; -0.71]$ ➔ **$\det(J_r(q^0)) = 0$**, **$J_r^T(q^0)*e = 0$**

  we have to choose another starting point like $q^1 = [0, \pi/6]$ ➔ $e = [-1.3\ 0.7]$ ➔ **$\det(Jr(q^0)) = 0$ but $J_r^T(q^0)*e = [1.4; 0.7] \neq 0$** so we can go on

  $q^2 = q^1 + \alpha J_r^T(q^1)e(q^1) = [0.67; 1.05]$ ➔ Using α = 0.7 ➔ **21** iterations are needed to arrive at the solution, instead using α = 1 ➔ 17 iterations are needed to arrive at the solution.

Moreover, it is necessary to find another point from which to start to find other solutions also because perhaps the solution found could be incompatible with the limits of the manipulator joints.

DIFFERENTIAL KINEMATICS

It shows the relationships between motion (velocity) in joint space and motion (linear / angular velocity) in task space (e.g. Cartesian space).
The instantaneous velocity mappings can be obtained by temporal differentiation of the direct kinematics or in a geometric way, directly at the differential level.
v and ω are "vectors", namely are elements of vector spaces
they can be obtained as the sum of single contributions (in any order)
such contributions will be given by the single (linear or angular) joint velocities.
On the other hand, ϕ (and $\dot{\phi}$) is not an element of a vector space then a minimal representation of a sequence of two rotations is not obtained summing the corresponding minimal representations (accordingly, for their time derivatives)

$\omega \neq \dot{\phi}$

Well known: • Finite and infinitesimal translations (linear displacements) always commute • Finite rotations do not commute **but infinitesimal commute** (product between infitesimal rotation is null)

The time derivative of an elementary rotation matrix is: $R = S(\omega)R^T$, where $I + S(d\varphi) = \begin{bmatrix} 1 & -d\phi_Z & d\phi_Y \\ d\phi_Z & 1 & -d\phi_X \\ -d\phi_Y & d\phi_X & 1 \end{bmatrix}$

Robot Jacobian matrices

Analytical Jacobian

$\dot{r} = \frac{df_r(q)}{dq}\dot{q} = J_r(q)\dot{q}$

Geometric or basic Jacobian

$[v_E;\ \omega_E] = [J_L(q);\ J_A(q)]\dot{q} = J(q)\dot{q}$, where: $J_L(q)$ ($J_A(q)$) is the sum of the contribution to the linear (angular) E-E velocity due to $\dot{q}_i$, if the i-th joint is prismatic it contributes only to the linear speed ($z_{i-1}d_i$) while the revolutionary one contributes to both ($z_{i-1} \times (p_E - p_{i-1});\ z_{i-1}\dot{d}_i$)

The Jacobian matrix depends on the (current) configuration of the robot, **USEFULL RELATIONS**:

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \quad \boldsymbol{v}_\perp = \boldsymbol{J}\boldsymbol{J}^{\#}\boldsymbol{v} \quad \boldsymbol{R} = \boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{R}^{\boldsymbol{T}} \quad \boldsymbol{\omega} = \boldsymbol{T}(\boldsymbol{\varphi})\dot{\boldsymbol{\theta}} \quad \boldsymbol{J}(\boldsymbol{q}) = [\boldsymbol{I}\,\boldsymbol{0};\ \boldsymbol{0}\,\boldsymbol{T}(\boldsymbol{\varphi})]\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{q}) \quad \boldsymbol{J}^{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{R}^{\boldsymbol{u}} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}^{\boldsymbol{u}} \end{bmatrix}\boldsymbol{J},$$

To change the reference speed from RF0 to RFU

For $\boldsymbol{n}$ = # joint space, $\boldsymbol{m}$ = # dimension of task space then:

- $\boldsymbol{p(J(q))} = \boldsymbol{m}$ at q (**J(q) has max rank**, with $m \leq n$), the end-effector can be **moved in any direction of the task space** $\mathbb{R}^m$,
- $\boldsymbol{p(J(q))} < \boldsymbol{m}$ at q **there are directions forbidden** for the manipulator (not instantaneously!) (we are in a **singularity** configuration),
- $\boldsymbol{N(J(q))} \neq \{\}$ there are **non-zero** $\dot{q}$ **joint speeds** they **produce zero speed of the effector** ("self-movements") could be used for repositioning.

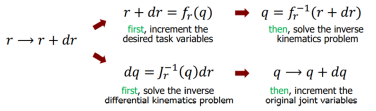
In a **singularity** there are **forbidden directions** instead **near** a **singularity** a **large joint speed** is required to make a **small displacement**, so finding and analyzing in advance the mobility of a robot helps in singularity avoidance during trajectory planning and motion control. But **finding all singular configurations** of a robot **with** a **large number of joints**, **or** the actual "**distance**" **from a singularity**, **is** a **complex computational task**.
In many cases, singularities separate configuration space regions with distinct inverse kinematic solutions (ex elbow up-down).

INVERSE DIFFERENTIAL KINEMATICS, STATICS AND FORCE TRANSFORMATIONS

To find the joint velocity vector that realizes a desired task/ end-effector velocity, $\dot{q} = J^{-1}(q)v$ we encounter problems if:

- We are close to a singularity (the speed values are too high)
- Using a redundant robot there is no inverse Jacobian

Joint velocity inversion can be used also to solve on-line and incrementally a "sequence" of inverse kinematics problems, each problem differs by a small amount *dr* from previous one.

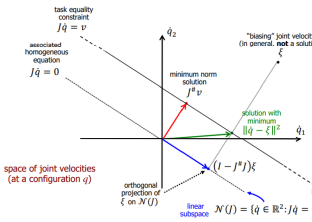


Example of two methods

To compute the J^{-1} of:

a. **Redundant manipulator** (or not) use $J^{-1} = J^{\#} = J^T(JJ^T)^{-1}$ (right pseudo-inverse of *J*) the solution locally **minimizes the norm of joint velocities**.
b. In singularity position is not possible invert J so we have to use the **Singular Value Decomposition** (in matlab pinv(J))
c. To get a $\dot{q}$ similar to $\dot{q}_0$ use $\dot{q} = J^{\#}v + (I - J^{\#}J)\dot{q}_0$ the second term is termed homogeneous solution cause $\in N(J(q))$ so $v = 0$ internal motions to reconfigure the posture of manipulator without changeing position of *E-E*

a simple case with $n = 2,\ m = 1$ at a given configuration $J\dot{q} = [j_1 \quad j_2]\begin{bmatrix}\dot{q}_1 \\ \dot{q}_2\end{bmatrix} = v \in \mathbb{R}$



- Higher-order differential inversion

Acceleration: $\ddot{q} = J_r^{-1}(q)(\ddot{r} - \dot{J}_r(q)\dot{q})$ **Jerk:** $\dddot{q} = J_r^{-1}(q)(\dddot{r} - \dot{J}_r(q)\ddot{q} - 2\ddot{J}_r(q)\dot{q})$

Used to obtain smoother motions

- τ are the forces/torques **exerted by the motors** at the robot joints
- F_e are the forces/torques **exerted by the environment** at the end-effector

The virtual work is the work done by all forces/torques acting on the system for a given virtual displacement, the sum is 0. $\tau = J^T(q)F$
The measure of the **velocity manipulability** is computed with $\dot{q}^T\dot{q} = 1 \rightarrow v^T J^{\#T} J^{\#} v = 1 \rightarrow \boldsymbol{v^T(JJ^T)^{-1}v = 1}$ is a elipsoid which the middle term is the **core** (kernel) the loss of a velocity dimension, is also usable a measure of the closeness to a singularity, is $\omega = \sqrt{\det(JJ^T)} = \prod \sigma_i = \prod \sqrt{\lambda_i(JJ^T)} \geq 0$, instead σ_i measure the distance of a singularity by the joint i.
Same for the **force manipulability** $\tau^T\tau = 1 \rightarrow F^T\boldsymbol{JJ^T}F = 1$

TRAJECTORY PLANNING

The trajectory can pass through precise points (a straight line) or use via points (interpolation of points).
After choosing a path, the trajectory definition is completed by the choice of a timing law:

- Cartesian space $p = p(s)$ which $s = s(t)$n if $s(t) = t$ the path parameterization is the natural one given by time.
- Joint space $q = q(\lambda),\ \lambda = \lambda(t)$

The timing law is based on task specifications constrints (bang-bang acceleration, costant velocity or determinated for some point ecc…)
Plan in **Cartesian space** let a more **direct visualization** of the generated **path** and **obstacle avoidance** instead planning in **Joint space** does **not need on-line kinematic inversion**.

Trajectories in Joint space:

📌 **Cubic polynomial**: (**4** conditions to define)

$q(0) = q_{in},\ q(T) = q_{fin},\ \dot{q}(0) = v_{in},\ \dot{q}(T) = v_{fin},\ \Delta q = q_{fin} - q_{in}$

$q(\tau) = q_{in} + \Delta q(a\tau^3 + b\tau^2 + c\tau + d), \tau = t/T\ (\sim \lambda) \in [0,1] \longrightarrow \dot{q}\left(\frac{1}{2}\right) = \dot{q}_{MAX}$

Special case for a rest-to-rest motion $v_{in} = v_{fin} = 0$

$\boldsymbol{a = -2, b = 3, c = 0, d = 0}$

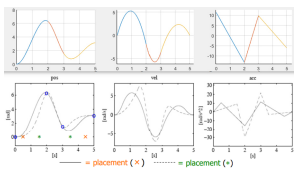
📌 **Quintic polynomial**: (**6** conditions to define)

$q(0) = q_i,\ q(T) = q_f,\ \dot{q}(0) = v_i T,$
$\dot{q}(T) = v_f T, \ddot{q}(0) = a_i T^2, \ddot{q}(T) = a_f T^2, \Delta q = q_f - q_i$

$q(\tau) = (1-\tau)^3 + (q_0 + (3q_0 + v_0 T)\tau + (a_0 T^2 + 6v_0 T + 12q_0)\tau^2/2)$
$+\tau^3(q_1 + (3q_1 - v_1 T)(1-\tau) + (a_1 T^2 - 6v_1 T + 12q_1)(1-t)^2/2)$
$\tau = t/T \in [0,1]$

Special case for a rest-to-rest motion of second order:
$v_i = v_f = a_i = a_f = 0$

$\boldsymbol{q(\tau) = q_i + \Delta q(6\tau^5 - 15\tau^4 + 10\tau^3)}$



Example of the effects to impose fiction dots in different time

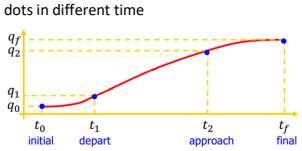
<u>High order of polynomials is usefull for satisfying symmetric boundary conditions that impose zero values on higher-order derivatives</u>
By interpolating N points with cubic polynomials, a spline is built and also having 2 other conditions free to set!

📌 **Spline**: (2 conditions free for N knots)
$\theta(t)$ depends on the interval time ($h_k = t_{k+1} - t_k$) so $\tau = t - t_k,\ \boldsymbol{\theta_k(\tau) = a_{k0} + a_{k1}\tau + a_{k2}\tau^2 + a_{k3}\tau^3}$, with the conditions of continuity (vel/acc.) for the internal knots: $\dot{\theta}_k(h_k) = \dot{\theta}_{k+1}(0), \ddot{\theta}_k(h_k) = \ddot{\theta}_{k+1}(0)\ k = 2 \ldots N-2$, if all the internal velocity is known impose:
$\theta_k(0) = q_k = a_{k0}, \dot{\theta}_k(0) = v_k = a_{k1}, \ddot{\theta}_k(h_k) = \ddot{\theta}_{k+1}(0) \rightarrow 2a_{k2} + 6a_{k3} = 2a_{k+1,2}$

- To **impose** the **initial and final acceleration** it's necessary to insert 2 fictitious knots therefore we have N+1 cubic polynomials with 2 free parameters in the first and last polynomial (they have as a condition only the continuity in the position, in the velocity and in the acceleration)

📌 **4-3-4 polynomials**: (costant velocity while travel)
By dividing the path into 3 pieces (lift off, travel and set down) you need to use three polynomials of degree 4-3-4 respectively in order to have a total of 14 coefficients to set:

$q(t_0) = q_0 \quad q(t_1^-) = q(t_1^+) = q_1 \quad q(t_2^-) = q(t_2^+) = q_2 \quad q(t_f) = q_f \quad \dot{q}(t_0) = \dot{q}(t_f) = 0 \qquad \ddot{q}(t_0) = \ddot{q}(t_f) = 0 \quad \dot{q}(t_i^-) = \dot{q}(t_i^+) \qquad \ddot{q}(t_i^-) = \ddot{q}(t_i^+) \qquad i = 1,2$



📌 For a large number of point to interpolate is not recommended, use of the unique **N-1 degree polynomial** which **interpolate N point** (have N-2 maximum/minimum = no wandering), on matlab compute with the function polyfit(t_k, q_k,N-1)

Trajectories in Cartesian space:

📌 **Bang-coast-bang**:(trapezoidal vel, reduce the time travel as function of the max vel/acc)

Knowing: L (lenght path), v_{max}, a_{max} $\quad \boldsymbol{T_s} = \frac{\boldsymbol{v_{max}}}{\boldsymbol{a_{max}}}, \boldsymbol{T} = \frac{\boldsymbol{La_{max}+v_{max}^2}}{\boldsymbol{a_{max}v_{max}}},$ $\quad$ a coast phase exist if $L > \frac{v_{max}^2}{a_{max}}$

$$\sigma(t) = \begin{cases} \frac{a_{max}t^2}{2}, t \in [0, T_s] \\ v_{max}t - \frac{v_{max}^2}{2a_{max}}, t \in [T_s, T - T_s] \\ -\frac{a_{max}(t-T)^2}{2} + v_{max}T - \frac{v_{max}^2}{a_{max}} \; t \in [T - T_s, T] \end{cases}$$

📌 **Via point**: (over-fly, costant acceleration)

Knowing: v1, v2, the points: A, B (will be overflown) and C

$$K_{AB} = \frac{B-A}{||B-A||}, K_{BC} = \frac{C-B}{||C-B||}$$

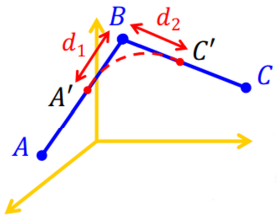
Imposing acceleration: $\ddot{p}(t) = \frac{v_2 K_{BC} - v_1 K_{AB}}{\Delta T} = a_{MAX}$

$$p(t) = A' + v_1 K_{AB} t + \frac{(v_2 K_{BC} - v_1 K_{AB})t^2}{2\Delta T}$$

$$d_1 = \frac{v_1 \Delta T}{2}, d_2 = \frac{v_2 \Delta T}{2}$$

$$\Delta T = \left(\frac{v_{MAX}}{a_{MAX}}\right)\sqrt{2(1 - K_{BC,x}K_{AB,x} - K_{BC,y}K_{AB,y} - K_{BC,z}K_{AB,z})}$$

$$d_1 = d_2 = \frac{v_{MAX}\Delta T}{2}$$



📌 **Axis/angle representation**: (Orientation trajectories)

$Rot(\theta_{if}) = R_i^T R_f$

$p(s) = p_i + s(p_f - p_i)$

$$\theta_{if} = atan2\left(\pm\sqrt{(R_{12}-R_{21})^2 + (R_{13}-R_{31})^2 + (R_{23}-R_{32})^2}, R_{11}+R_{22}+R_{33}-1\right)$$

$$r = 1/(2\,sin\theta)\begin{bmatrix} R_{32}-R_{23} \\ R_{13}-R_{31} \\ R_{21}-R_{12} \end{bmatrix}$$

$R(s) = R_i Rot(\theta(s))$

$\theta(s) = s\theta_{if}$

📌 **Uniforming time scaling**: (increase time of k times to remain in constrints vel/acc iff violated)

Velocity scale linearly with motion time instead acceleration scale quadratically.

$$K_{vel} = \max\left\{1, \frac{|\dot{q}_i|}{v_{max,i}}\right\}\forall i, \quad K_{acc} = \max\left\{1, \frac{|\ddot{q}_i|}{v_{acc,i}}\right\}\forall i \qquad k = \max\{1, k_{vel}, \sqrt{k_{acc}}\}$$

KINEMATIC CONTROL

TO DO…

Do you want to **help other guys** by **finishing** this **summary** or **correcting** the **errors found**?
Send an e-mail to **disabatino.2015738@studenti.uniroma1.it** or **ntonio.1997@gmail.com.**
Thanks for the feedback!