Humanoid robots - Kinematics II
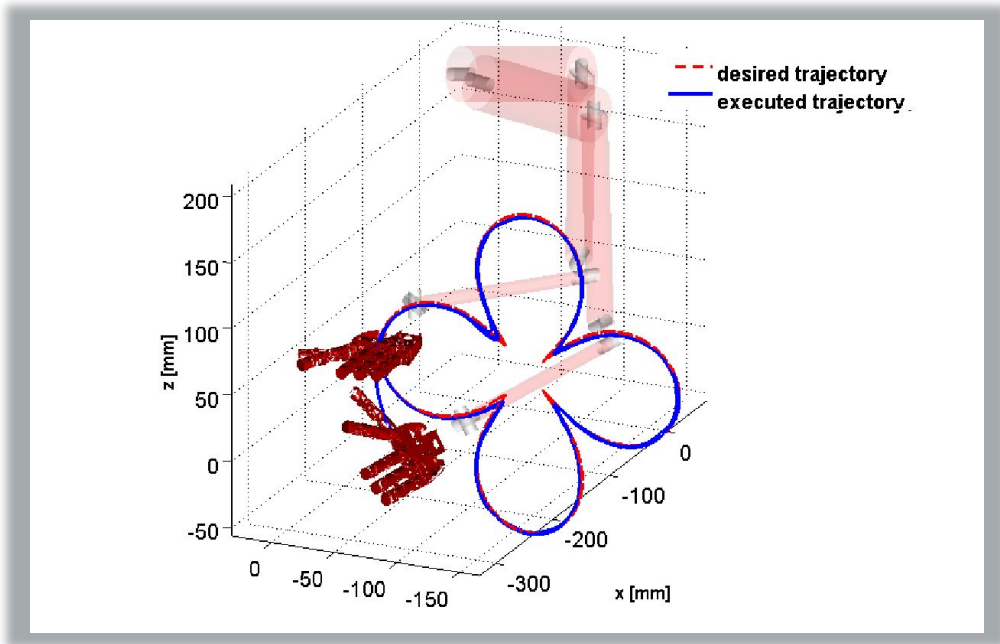
Doc. Mgr. Matěj Hoffmann, Ph.D.

Kinematics - forward and inverse

Study of properties of motion (position, velocity, acceleration) without considering body inertias and internal/external forces.
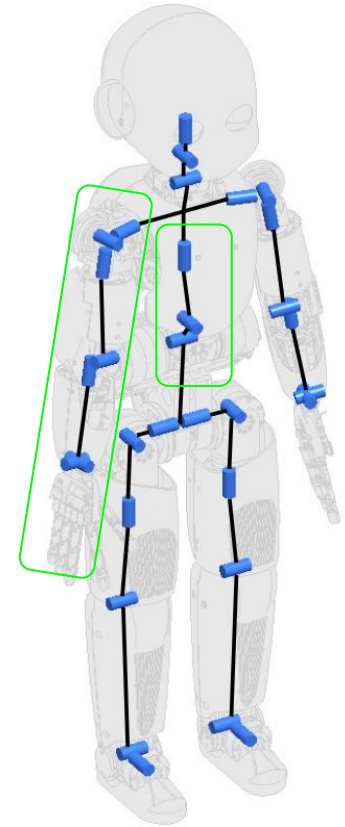
The Problem

$$\begin{cases} \mathbf{x} = \mathbf{f}(\mathbf{q}) \\ \mathbf{q} \in \mathbb{R}^n \\ \mathbf{x} \in \mathbb{R}^6 \end{cases}$$

$$\mathbf{q} \overset{?}{=} \mathbf{f}^{-1}(\mathbf{x})$$

Slide source:

Kinematic redundancy

- Whenever the limb mobility, determined by the number of limb joints n, exceeds the DoFs of the end link (six), the limb is characterized as kinematically redundant.
- Some humanoid robots are equipped with kinematically redundant, 7-DoF arms. Such robots can control the position of their elbows without affecting thereby the instantaneous motion of the hands. Thus, they attain the capability to perform tasks in cluttered environments avoiding collisions with their elbows, similar to humans.
- Also, there are humanoids that comprise 7-DoF legs. With proper control, their gait appears more human-like than that of robots with 6-DoF legs.
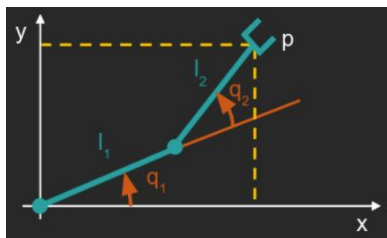- The difference r = n − 6 is referred to as the *degree of redundancy* (DoR).

Section 2.7 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.

https://youtu.be/sZYBC8Lrmdo

Forward and inverse kinematics – recap

Consider a general case: $\mathbf{q} \in \mathbb{R}^n$ $\quad$ $\mathbf{x} \in \mathbb{R}^m$

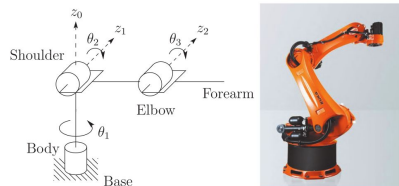n=7, m=6 (x,y,z, ɸ, θ, ψ)

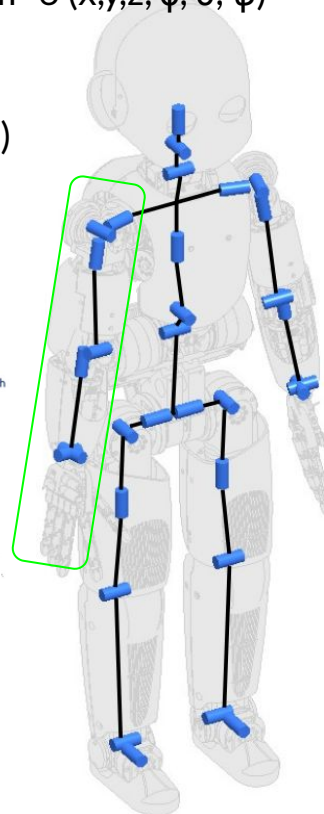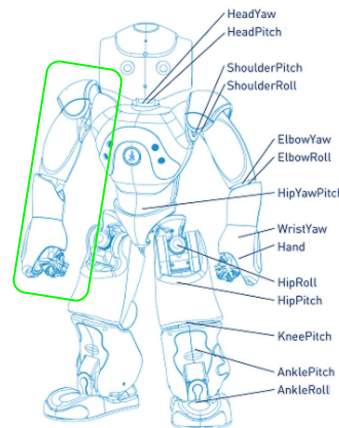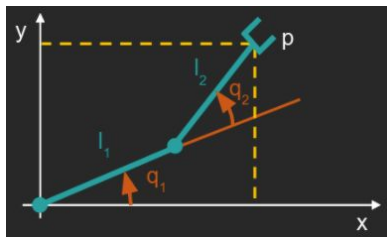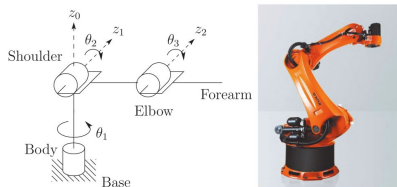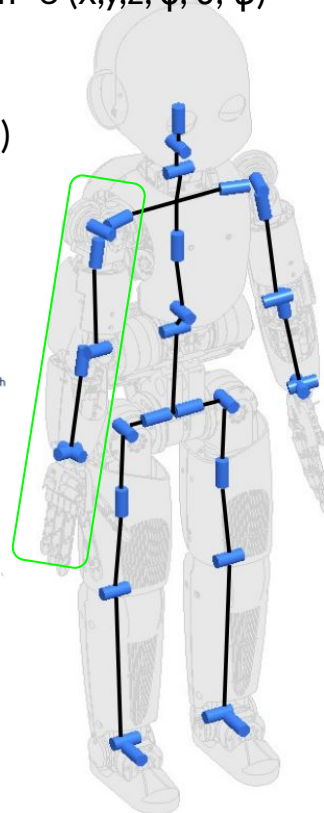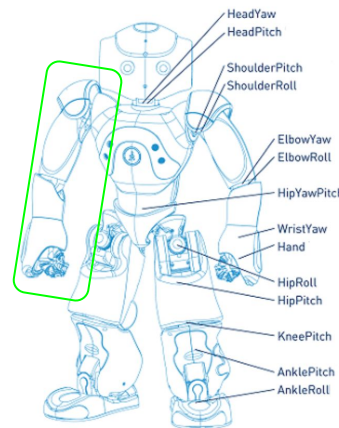n=2, m=3 (x,y,ɸ)   n=6, m=6 (x,y,z, ɸ, θ, ψ)   n=5, m=6 (x,y,z, ɸ, θ, ψ)

What is n and m?

	Mapping type	linear / nonlinear
$\mathbf{x} = \mathbf{f(q)}$		
$\mathbf{q} = \mathbf{f}^{-1}(\mathbf{x})$		

Forward and inverse kinematics – recap

n=7, m=6 (x,y,z, φ, θ, ψ)

Consider a general case: $\mathbf{q} \in \mathbb{R}^n \qquad \mathbf{x} \in \mathbb{R}^m$

n=2, m=3 (x,y,φ)   n=6, m=6 (x,y,z, φ, θ, ψ)   n=5, m=6 (x,y,z, φ, θ, ψ)

What is n
and m?

	Mapping type	linear / nonlinear
$\mathbf{x} = \mathbf{f}(\mathbf{q})$	Many-to-one (for n>= m)	nonlinear
$\mathbf{q} = \mathbf{f}^{-1}(\mathbf{x})$	One-to-many (for n>1)	nonlinear

Inverse kinematics – analytic (closed-form) solutions

$$\begin{cases} \mathbf{x} = \mathbf{f}(\mathbf{q}) \\ \mathbf{q} \in \mathbb{R}^n \\ \mathbf{x} \in \mathbb{R}^6 \end{cases}$$

$$\mathbf{q} = \mathbf{f}^{-1}(\mathbf{x})$$

TABLE 1
Number of Analytic Solutions for
6-Degree-of-Freedom Systems

n	Upper bound on solutions
<6	
>6	
6R, 5RP	
4R2P, 6R with S joint	
3R3P	

Tolani, D., Goswami, A., & Badler, N. I. (2000). Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, *62*(5), 353-388.

Inverse kinematics – analytic (closed-form) solutions

$$\begin{cases} \mathbf{x} = \mathbf{f}(\mathbf{q}) \\ \mathbf{q} \in \mathbb{R}^n \\ \mathbf{x} \in \mathbb{R}^6 \end{cases}$$

$$\mathbf{q} = \mathbf{f}^{-1}(\mathbf{x})$$

TABLE 1

Number of Analytic Solutions for 6-Degree-of-Freedom Systems

n	Upper bound on solutions
<6	0
>6	∞
6R, 5RP	16
4R2P, 6R with S joint	8
3R3P	2

Tolani, D., Goswami, A., & Badler, N. I. (2000). Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, *62*(5), 353-388.

Inverse kinematics – closed form

- fast at runtime
- suited for platforms designed with IK solutions in mind (joint alignment - e.g. spherical wrist)
- difficult for "general" or redundant robots
  - laborious to develop a solution
  - complicated to incorporate constraints, especially dynamic constraints
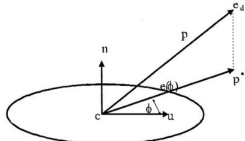- does not provide trajectories (motion control)

Wrist position and E-E pose
Inverse solutions for an articulated 6R robot [PUMA 560]

LEFT UP

RIGHT DOWN

4 inverse solutions
(for the position of the wrist center only)

8 inverse solutions
considering the complete E-E pose (spherical wrist: 2 alternative solutions for the last 3 joints)

LEFT UP

RIGHT UP

TABLE 1
Number of Analytic Solutions for
6-Degree-of-Freedom Systems

n	Upper bound on solutions
<6	0
>6	∞
6R, 5RP	16
4R2P, 6R with S joint	8
3R3P	2

FIG. 3.   Finding the elbow position that is the closest possible to a desired position.

TABLE 2
Summary of Methods Used When the Goal Is Reachable

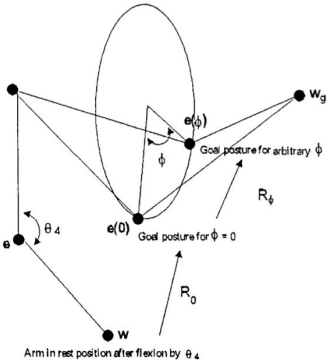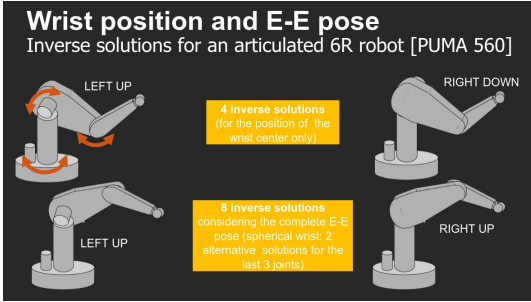|	Goal reachable (joint limits off)	Goal reachable (joint limits on)
Position	Analytic	Analytic
Position and orientation	Analytic	Analytic
Position and partial orientation	Analytic	Analytic + 2DOF unconstrained optimization
Aiming	Analytic	Analytic if θ_4 given 2DOF unconstrained optimization otherwise

FIG. 5.   Decomposing R_1 into R_0 and R_ϕ.
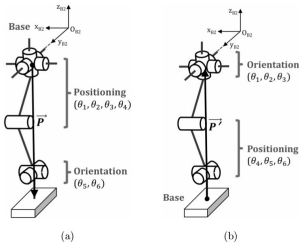

Fig. 5.   (a) Forward Decoupling of the Right Leg of a Hubo KHR-4 Robot, (b) Reverse Decoupling of the Right Leg of a Hubo KHR-4 Robot.
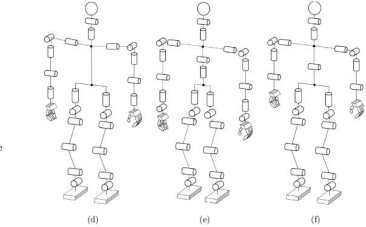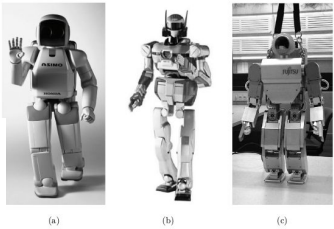
Fig. 4.   (a) HONDA ASIMO Robot and its associated kinematic diagram in (d), (b) AIST HRP-2 Robot and its associated kinematic diagram in (e), and (c) Fujitsu HOAP-2 Robot and its associated kinematic diagram in (f).

Tolani, D., Goswami, A., & Badler, N. I. (2000). Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, *62*(5), 353-388.

Park, H. A., Ali, M. A., & Lee, C. G. (2012). Closed-form inverse kinematic position solution for humanoid robots. *International Journal of Humanoid Robotics*, *9*(03), 1250022.

Inverse kinematics – iterative method

- iCub Cartesian solver

$$\tilde{q}_d = \arg \min_{q \in \mathbb{R}^n} \left(\left\| \alpha_d - K_\alpha (q) \right\|^2 + \lambda \cdot \left(q_{\text{rest}} - q \right)^T W \left(q_{\text{rest}} - q \right) \right)$$

$$\text{s.t.} \begin{cases} \left\| x_d - K_x (q) \right\|^2 < \varepsilon \\ q_L < q < q_U \\ \text{other obstacles ...} \end{cases}$$

➢ **Quick convergence**: real-time compliant, < **20 ms**
➢ **Scalability**: n can be high and set on the fly
➢ **Singularities handling**: no Jacobian inversion
➢ **Joints bound handling**: no explicit boundary functions
➢ **Tasks hierarchy**: no use of null space
➢ **Complex constraints**: intrinsically nonlinear

https://github.com/vvv-school/vvv18/blob/master/material/kinematics/kinematics.pdf, courtesy Ugo Pattacini

From joint positions to joint velocities

$(x, y, z, \varphi, \theta, \psi)$ → **Solver module** → q_d → **Controller module** → $\dot{q}$

q
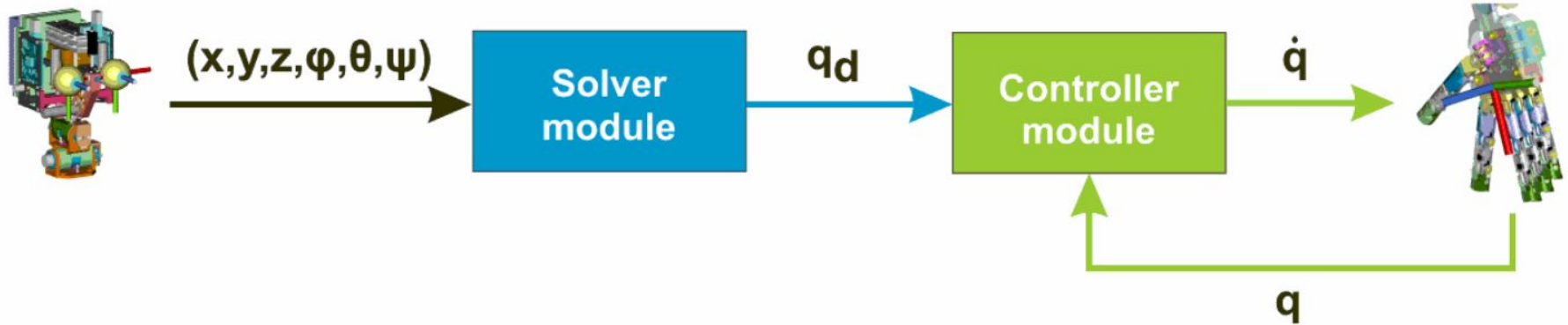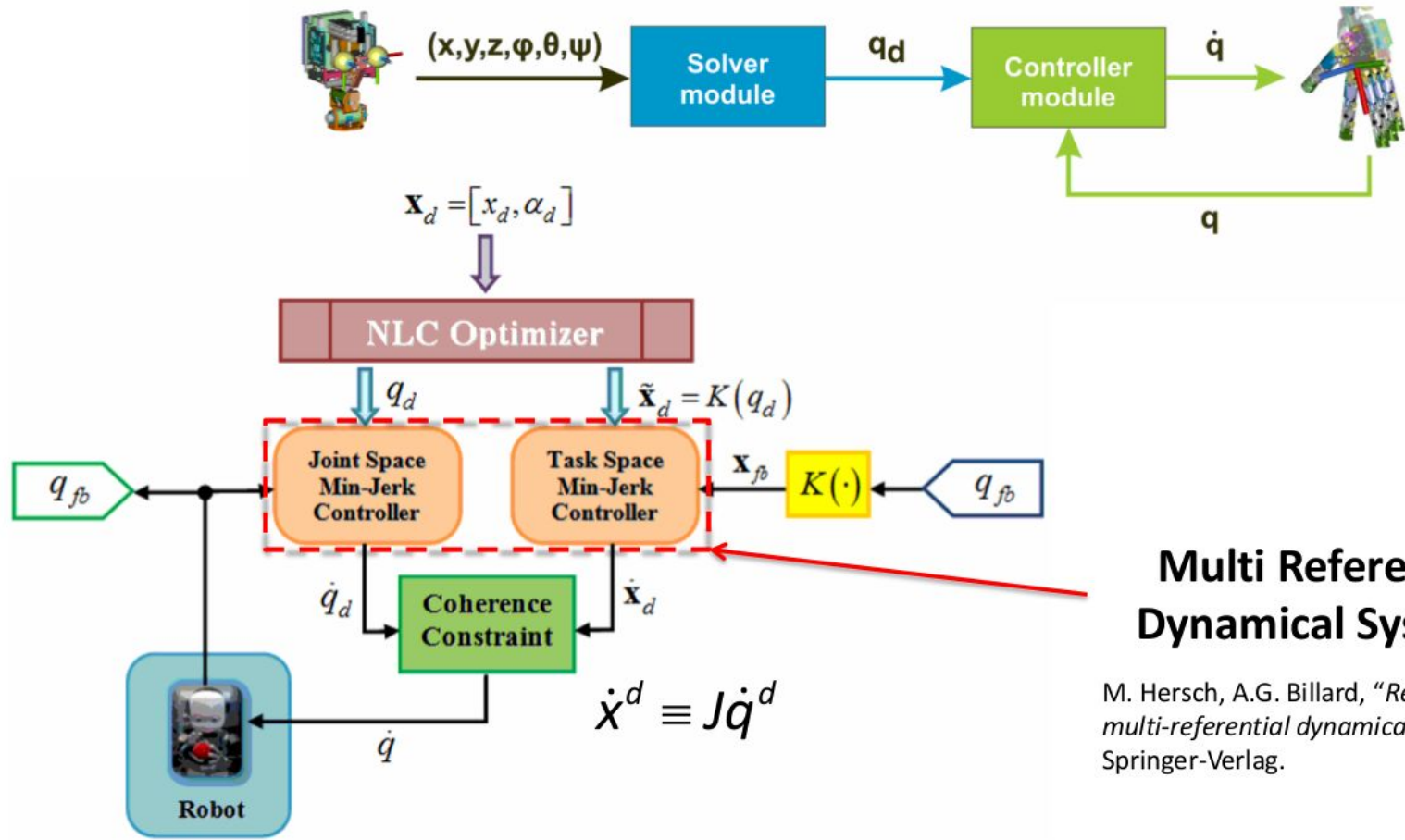
Multi Referential Dynamical Systems

M. Hersch, A.G. Billard, "*Reaching with multi-referential dynamical systems*", Springer-Verlag.

$$\dot{x}^d \equiv J\dot{q}^d$$

Other stakeholders

- Who else is interested in inverse kinematics of anthropomorphic creatures?
- Computer graphics / animation / game industry!

Other stakeholders

- Who else is interested in inverse kinematics of anthropomorphic creatures?
- Computer graphics / animation / game industry!

https://youtu.be/0a9qIj7kwiA?t=321

https://youtu.be/SHpImEc6iv0?t=156
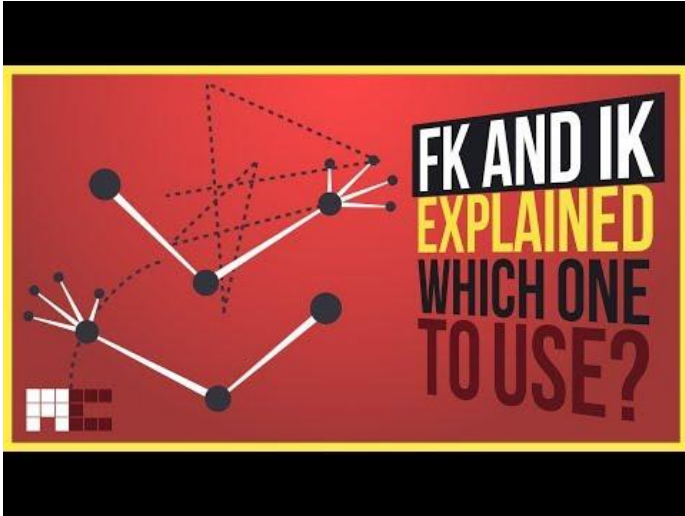
Inverse kinematics – robotics vs. animations

Humanoid robots ~ human-like characters in games etc.

Which problem is harder?

Constraints seem less strict for the animation industry:

- joint limits (position, velocity…)
- self-collisions
- [balance]

Differential kinematics

- Velocity relationships relating linear and angular velocities of the end effector to the joint velocities.
- "Mathematically, the forward kinematic equations define a function from the configuration space of joint positions to the space of Cartesian positions and orientations." (Spong et al., pg. 101)
- "The velocity relationships are then determined by the **Jacobian** of this function. The Jacobian is a matrix that generalizes the notion of the ordinary derivative of a scalar function. The Jacobian is one of the most important quantities in the analysis and control of robot motion. It arises in virtually every aspect of robotic manipulation: in the planning and execution of smooth trajectories, in the determination of singular configurations, in the execution of coordinated anthropomorphic motion, in the derivation of the dynamic equations of motion, and in the transformation of forces and torques from the end effector to the manipulator joints." (Spong et al., pg. 101)
- Today: iterative methods for IK, singularity and manipulability.
- Basics of Jacobian, manipulability etc. covered in Robotics by V. Smutný (B3B33ROB1).

Chapter 4 in Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2020). *Robot modeling and control*. John Wiley & Sons.

Geometrical Jacobian

$$\nu = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \omega_z}{\partial q_1} & \frac{\partial \omega_z}{\partial q_2} & \cdots & \frac{\partial \omega_z}{\partial q_n} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} = J(q) \cdot \dot{q}$$

twist

- it is a function of the joint configuration q
- contains all of the partial derivatives of f , relating every joint angle to every velocity
- tells us how small changes in joint space will affect the end-effector's position in Cartesian space
- columns: how each component of velocity changes when the configuration (i.e., angle) of a particular joint changes
- rows: how movement in each joint affects a particular component of velocity

Analytical Jacobian

$$\mathbf{x} = \mathbf{f}(\mathbf{q}) \overset{d/dt}{\Longrightarrow} \dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \dfrac{\partial f_1}{\partial q_1} & \cdots & \dfrac{\partial f_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial q_1} & \cdots & \dfrac{\partial f_m}{\partial q_n} \end{bmatrix}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_P \\ \mathbf{J}_\phi \end{bmatrix} \dot{\mathbf{q}}$$

Slide source: https://github.com/vvv-school/vvv18/blob/master/material/kinematics/kinematics.pdf. Courtesy Ugo Pattacini.

Planar arm with 2 rotational joints – forward kinematics (position only)

$$x_1 = L_1 \cos \theta_1 + L_2 \cos(\theta_1 + \theta_2)$$
$$x_2 = L_1 \sin \theta_1 + L_2 \sin(\theta_1 + \theta_2).$$

Ch.5 Velocity kinematics and statics in Lynch, K. M., & Park, F. C. (2017). Modern robotics. Cambridge University Press.
(see also https://youtu.be/6tj8QLF69Ok)

Planar arm with 2 rotational joints – forward differential kinematics (position only)

$$
\begin{aligned}
x_1 &= L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) \\
x_2 &= L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2).
\end{aligned}
$$

Differentiating both sides with respect to time yields

$$
\begin{aligned}
\dot{x}_1 &= -L_1 \dot{\theta}_1 \sin\theta_1 - L_2(\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_1 + \theta_2) \\
\dot{x}_2 &= L_1 \dot{\theta}_1 \cos\theta_1 + L_2(\dot{\theta}_1 + \dot{\theta}_2)\cos(\theta_1 + \theta_2),
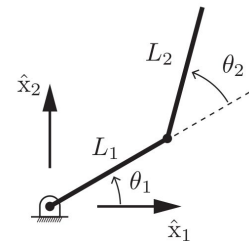\end{aligned}
$$

which can be rearranged into an equation of the form $\dot{x} = J(\theta)\dot{\theta}$:

$$
\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -L_1\sin\theta_1 - L_2\sin(\theta_1 + \theta_2) & -L_2\sin(\theta_1 + \theta_2) \\ L_1\cos\theta_1 + L_2\cos(\theta_1 + \theta_2) & L_2\cos(\theta_1 + \theta_2) \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}. \quad (5.1)
$$

Writing the two columns of $J(\theta)$ as $J_1(\theta)$ and $J_2(\theta)$, and the tip velocity $\dot{x}$ as v_{tip}, Equation (5.1) becomes

$$
v_{\text{tip}} = J_1(\theta)\dot{\theta}_1 + J_2(\theta)\dot{\theta}_2. \quad (5.2)
$$

$J_1(\theta) + J_2(\theta)$: basis for the linear velocities of the end effector (with coefficients equal to joint velocities)

end effector velocity when joint 1 rotates at unit speed (joint 2 is still)

Ch.5 Velocity kinematics and statics in Lynch, K. M., & Park, F. C. (2017). Modern robotics. Cambridge University Press.
(see also https://youtu.be/6tj8QLF69Ok)
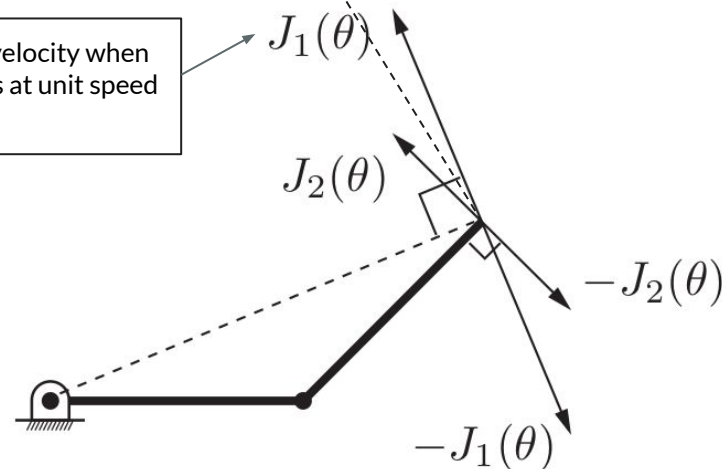
Straightforward generalization to 3D
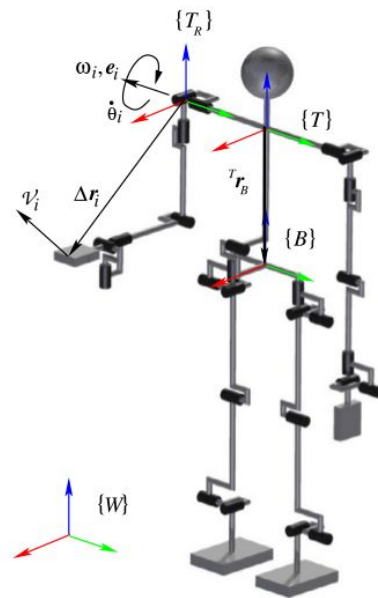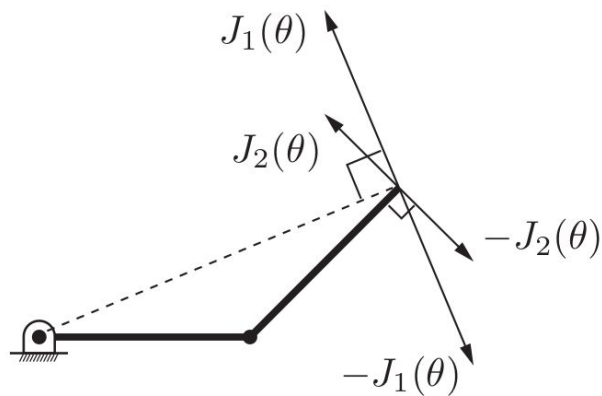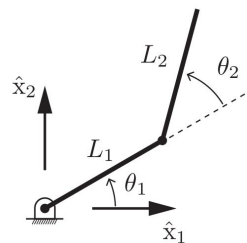
FIGURE 2.3 End-link spatial velocity $\mathcal{J}_i = [(e_i \times r_i)^T \quad e_i^T]^T$ is obtained with joint rate $\dot{\theta}_i = 1$ rad/s. Vector e_i signifies the joint axis of rotation. The position r_i of the characteristic point on the end link is determined w.r.t. reference frame $\{T_R\}$, obtained by translating the common root frame for the arms, $\{T\}$, to a suitably chosen point on the joint axis, e.g. according to the Denavit and Hartenberg notation [26].

Section 2.4 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.

iCub matlab demo
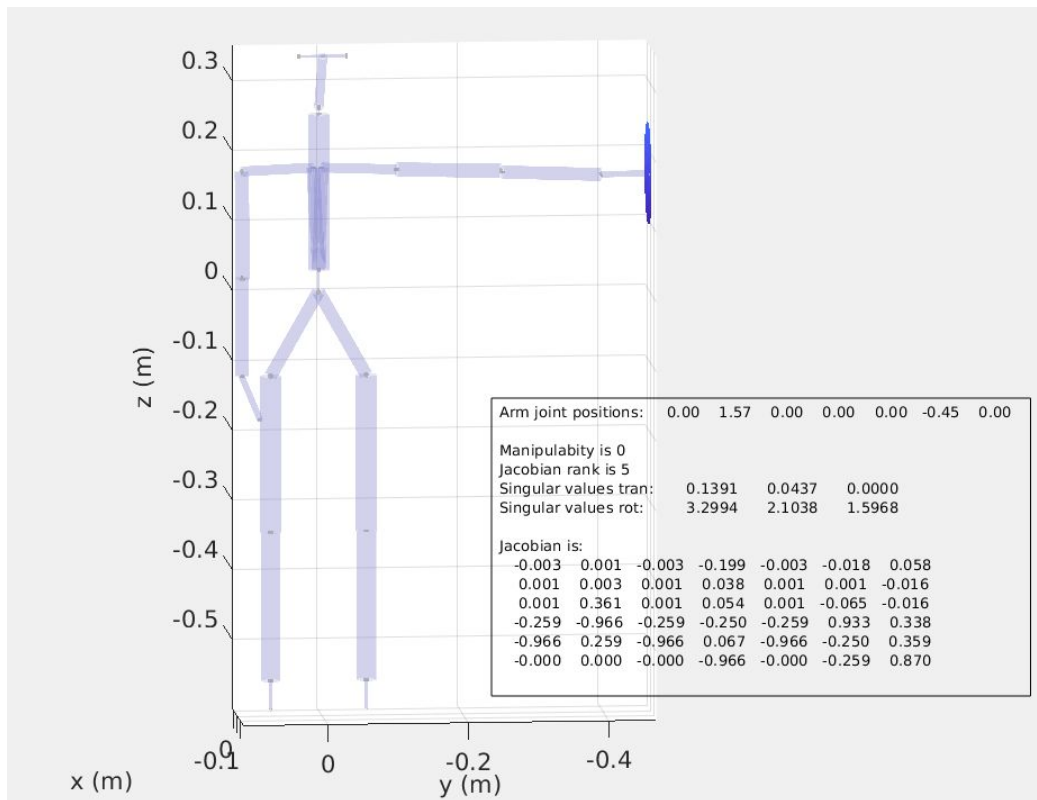
Planar arm with 2 rotational joints – singularities (position only)

$$x_1 = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$
$$x_2 = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2).$$

Differentiating both sides with respect to time yields

$$\dot{x}_1 = -L_1\dot{\theta}_1 \sin\theta_1 - L_2(\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_1 + \theta_2)$$
$$\dot{x}_2 = L_1\dot{\theta}_1 \cos\theta_1 + L_2(\dot{\theta}_1 + \dot{\theta}_2)\cos(\theta_1 + \theta_2),$$

which can be rearranged into an equation of the form $\dot{x} = J(\theta)\dot{\theta}$:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -L_1 \sin\theta_1 - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) \\ L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}. \quad (5.1)$$

Writing the two columns of $J(\theta)$ as $J_1(\theta)$ and $J_2(\theta)$, and the tip velocity $\dot{x}$ as v_{tip}, Equation (5.1) becomes

$$v_{tip} = J_1(\theta)\dot{\theta}_1 + J_2(\theta)\dot{\theta}_2. \quad (5.2)$$

- $\theta_2 = 0°$ (or 180°)
- $J_1(\theta)$ and $J_2(\theta)$ aligned
- impossible to generate any end effector velocity except along this line
- dimension of the column space of the Jacobian drops from its maximum value

Ch.5 Velocity kinematics and statics in Lynch, K. M., & Park, F. C. (2017). Modern robotics. Cambridge University Press.
(see also https://youtu.be/6tj8QLF69Ok)

23
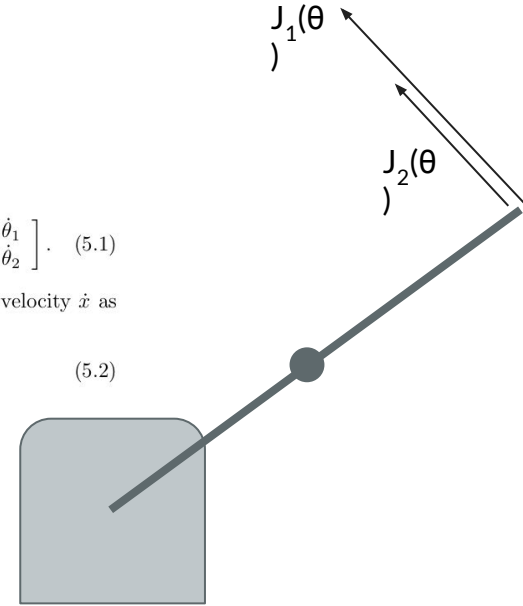
Differential kinematics at singular configurations

- There are certain limb configurations where the end link loses mobility, i.e. the ability of instantaneous motion in one or more directions.
- At singularities, bounded end-effector velocities may correspond to unbounded joint velocities.
- Since the arms are fully extended, the hands cannot move in the downward direction w.r.t. the {T} frame.
- Since the legs are stretched, the {B} frame cannot be moved in the upward direction.
- Elbow / knee singularities - unavoidable.
  - There are no alternative nonsingular configurations that would place the end links at the same locations, at the workspace boundaries of each limb.
  - Inherent to both redundant and nonredundant limbs.

Section 2.5 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.

Differential kinematics at singular configurations

rotate
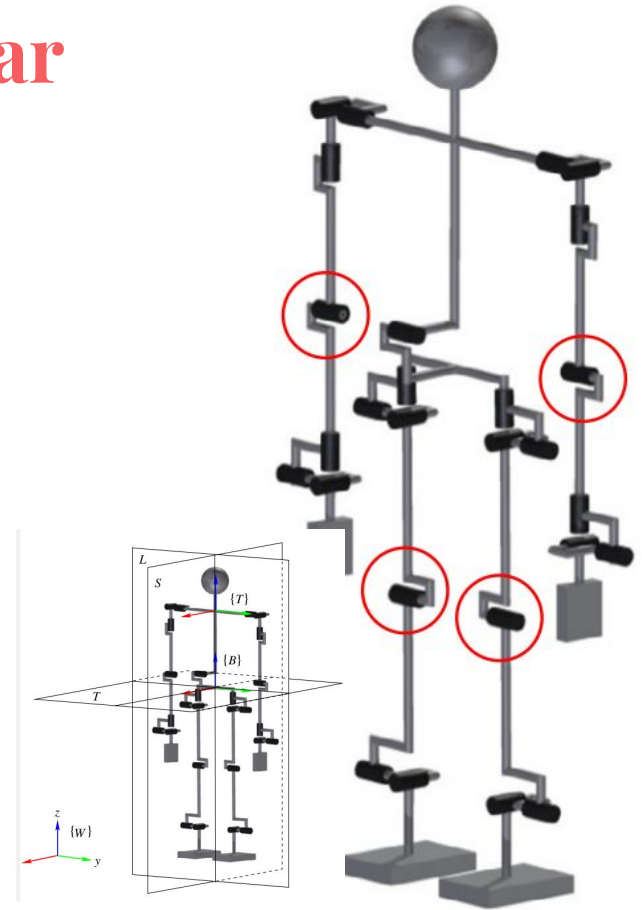
- "Fig. 2.4C shows another type of singular configuration for the right arm. The singularity is due to the alignment of the two axes in the shoulder joint, the elbow joint being at 90 degrees. The end link loses mobility in the translational direction of the lower-arm link. This configuration is called *shoulder singularity*. Note that the end link is placed within the workspace; it is not on the boundary. In this case, the self-motion of the arm, i.e. a motion whereby the end link is fixed, yields a transition to a nonsingular configuration. Such types of singular configurations are characterized as *avoidable*."

see also https://robohub.org/3-types-of-robot-singularities-and-how-to-avoid-them/

Section 2.5 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.
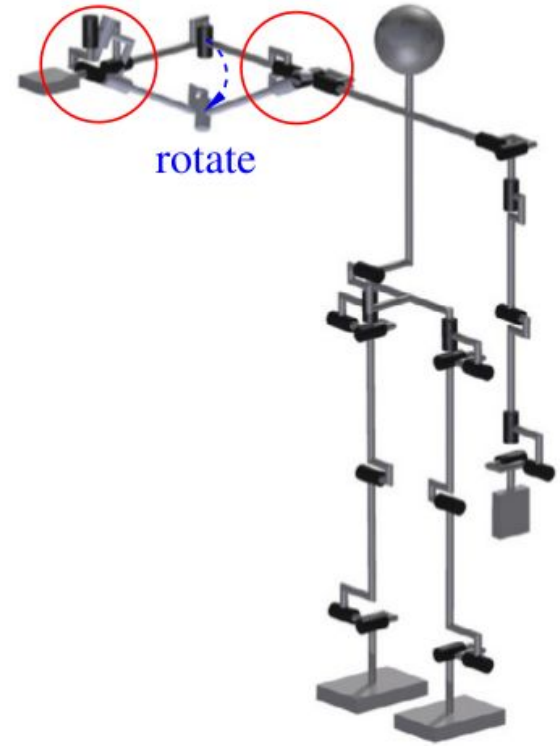
Differential kinematics at singular configurations

- Singularities can be also useful though! When?
- Resisting external forces with minimal load in the joints.
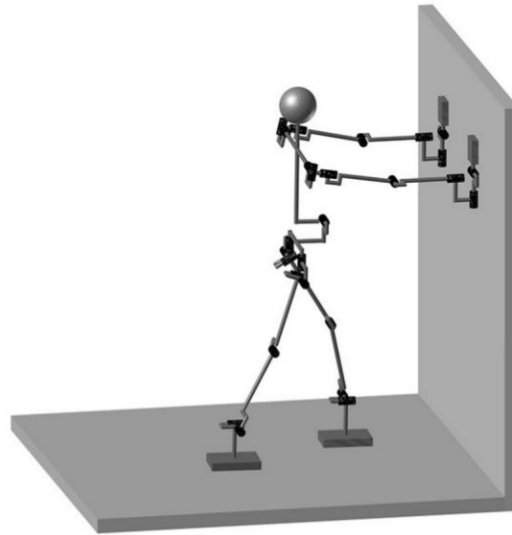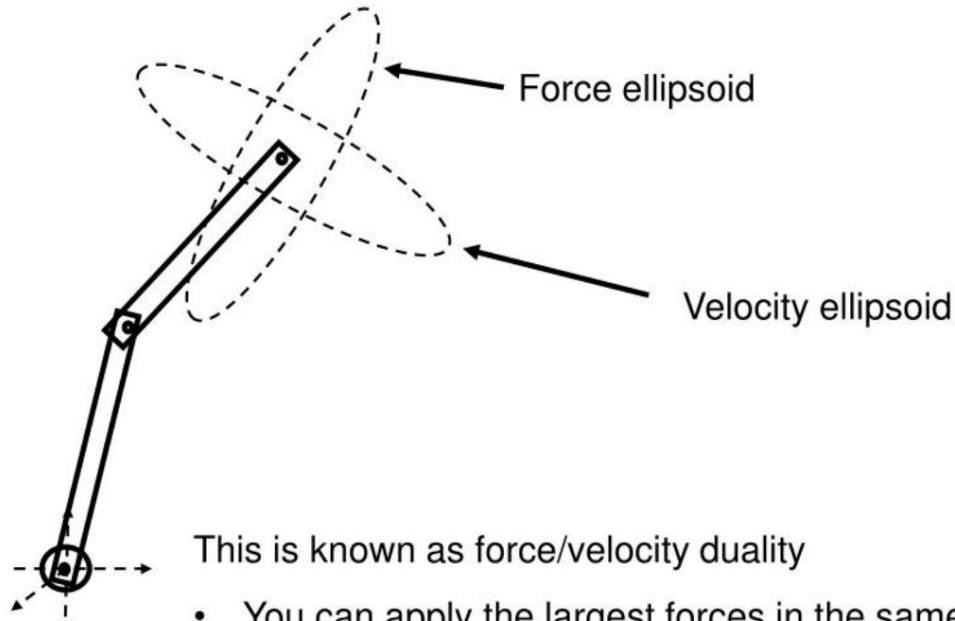
Section 2.5 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.

Velocity and force manipulability are orthogonal!

Force ellipsoid

Velocity ellipsoid

This is known as force/velocity duality

- You can apply the largest forces in the same directions that your max velocity is smallest

- Your max velocity is greatest in the directions where you can only apply the smallest forces

Slide 40 https://www.slideserve.com/antonia/inverting-the-jacobian-and-manipulability.
See slides 38-39 for details how this is derived.

Manipulability

2.6 MANIPULABILITY ELLIPSOID

From the differential kinematics relation (2.11), it is apparent that the ability of the end link to move instantaneously along a given spatial (rigid-body motion) direction will depend on the current limb configuration. In particular, as already clarified, at a singular configuration the ability to move along the singular directions becomes zero, and hence, mobility is lost in these directions. To facilitate instantaneous motion analysis and control, it is quite desirable to quantify the mobility in a given direction, at any given configuration. This can be done via *Singular-Value Decomposition* (SVD) [42,147,90] of the Jacobian matrix. For the general case of an n-DoF kinematically redundant limb, we have

$$J(\theta) = U(\theta)\Sigma(\theta)V(\theta)^T, \tag{2.26}$$

where $U(\theta) \in \Re^{6\times 6}$ and $V(\theta) \in \Re^{n\times n}$ are orthonormal matrices and

$$\Sigma(\theta) = \begin{bmatrix} \mathrm{diag}\{\sigma_1(\theta), \ \sigma_2(\theta), ..., \sigma_6(\theta)\} & | & 0 \end{bmatrix} \in \Re^{6\times n}. \tag{2.27}$$

Here $\sigma_1 \geq \sigma_2 \geq, ..., \geq \sigma_6 \geq 0$ are the singular values of the Jacobian. The columns of matrix $U(\theta)$, u_i, $i = 1, ..., 6$, provide a basis for the instantaneous motion space of the end link at the given limb configuration. At a nonsingular limb configuration, all singular values are positive. At a singular configuration of corank $6 - \rho$ ($\rho = \mathrm{rank}\, J$), $6 - \rho$ of the singular values become zeros, i.e. $\sigma_1 \geq \sigma_2 \geq, ..., \geq \sigma_\rho > 0$, $\sigma_{\rho+1} = ... = \sigma_6 = 0$. The singular value σ_i quantifies the instantaneous mobility of the end link along the instantaneous motion direction u_i. Assuming that the magnitude of the joint rate vector is limited at each limb configuration as $\|\dot{\theta}\| \leq 1$, the highest mobility is along the direction corresponding to the maximum singular value. At a singular configuration of corank 1, $\sigma_{min} = 0$ and the respective direction u_{min} becomes a singular direction. Vectors $\sigma_i u_i$ constitute the principal axis of an ellipsoid—a useful graphic tool for visualizing the instantaneous mobility along each possible motion direction.

28

Manipulability ellipsoid

not isotropic ✖

- Unit circle of joint velocities maps through the Jacobian to an ellipse in the space of tip velocities - the *manipulability ellipsoid*.
- As the manipulator configuration approaches a singularity, the ellipse collapses to a line segment, since the ability of the tip to move in one direction is lost.
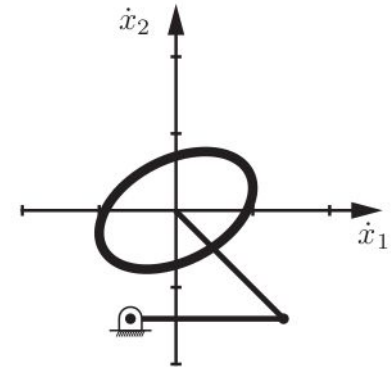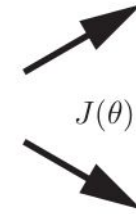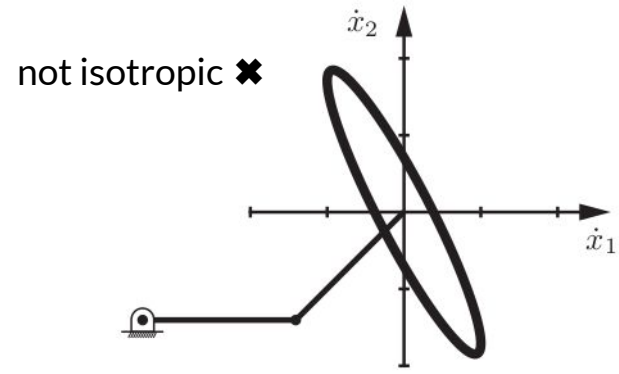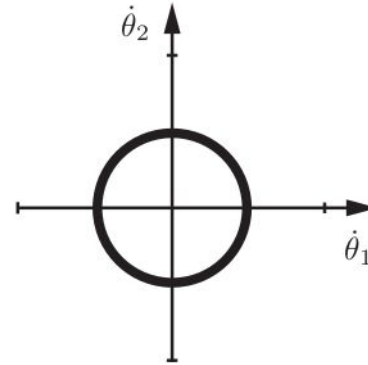
Unit circle of joint velocities in the $\theta_1 - \theta_2$ -plane.
- "iso-effort" contour in the joint velocity space, where total actuator effort is considered to be the sum of squares of the joint velocities

isotropic ✔

Ch.5 Velocity kinematics and statics in Lynch, K. M., & Park, F. C. (2017). Modern robotics. Cambridge University Press. (see also https://youtu.be/6tj8QLF69Ok)

Manipulability ellipsoid
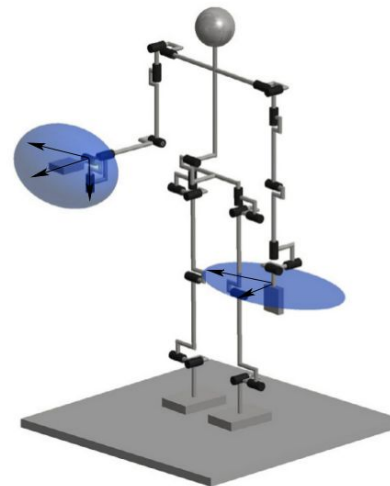
FIGURE 2.8 Manipulability ellipsoid for translational motion. The right arm is in a nonsingular configuration and the respective ellipsoid is 3D, with principal axes $\sigma_1 u_1$, $\sigma_2 u_2$, and $\sigma_3 u_3$. The left arm is at a singular configuration: the downward translational mobility has been lost, and therefore, the manipulability ellipsoid is only 2D. The principal axes are $\sigma_1 u_1$ and $\sigma_2 u_2$.

The dimension of the ellipsoid is determined by the rank of the Jacobian. Fig. 2.8 shows a robot configuration wherein the right arm is at a nonsingular configuration, whereas the left one is at the elbow singularity. The two ellipsoids at the end links visualize the instantaneous translational motion abilities. The ellipsoid for the right arm is 3D (full translational mobility), while that for the left arm is flat (an ellipse). The ellipse lies in a plane parallel to the floor since translational mobility in the vertical direction is nil at the singularity. The ellipsoid-based instantaneous mobility analysis has been introduced in [166]; the ellipsoid is referred to as the *manipulability ellipsoid*.

Section 2.6 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.

iCub matlab demo


```
Arm joint positions:   0.00   1.57   0.00   0.00   0.00  -0.45   0.00

Manipulabity is 0
Jacobian rank is 5
Singular values tran:    0.1391    0.0437    0.0000
Singular values rot:     3.2994    2.1038    1.5968

Jacobian is:
 -0.003   0.001  -0.003  -0.199  -0.003  -0.018   0.058
  0.001   0.003   0.001   0.038   0.001   0.001  -0.016
  0.001   0.361   0.001   0.054   0.001  -0.065  -0.016
 -0.259  -0.966  -0.259  -0.250  -0.259   0.933   0.338
 -0.966   0.259  -0.966   0.067  -0.966  -0.250   0.359
 -0.000   0.000  -0.000  -0.966  -0.000  -0.259   0.870
```

See also:

- Vahrenkamp, N., Asfour, T., Metta, G., Sandini, G., & Dillmann, R. (2012, November). Manipulability analysis. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)* (pp. 568-573). IEEE.
- https://github.com/robotology/community/discussions/559#:~:text=Ipopt%20doesn%27t%20deal,Jacobian%20per%20se

Using J⁻¹ for Cartesian control
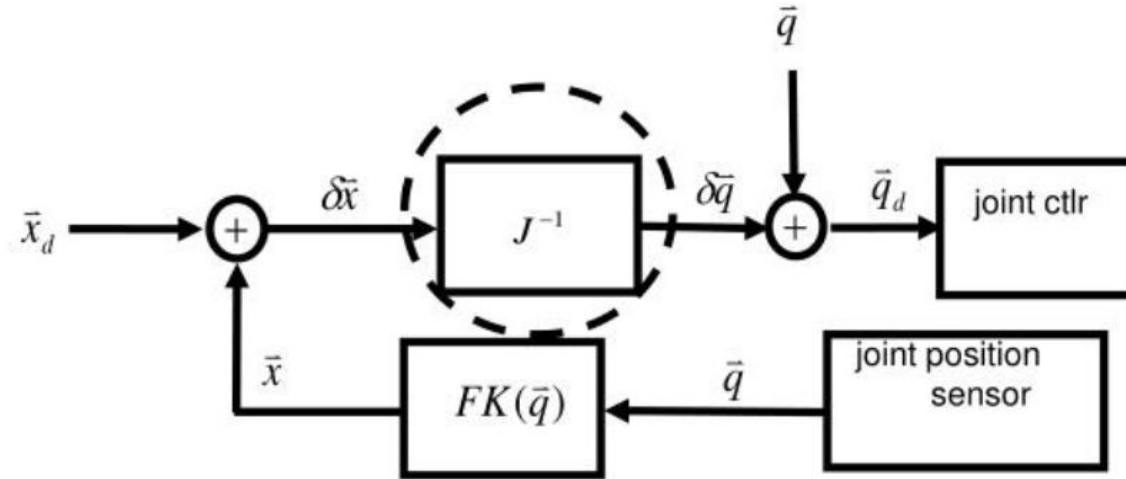
$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = J^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

Underactuation, full actuation, and over−actuation

Fig. 8.6 in Corke, P. I. (2013). Robotics, vision and control: fundamental algorithms in MATLAB Berlin: Springer.

$$\dot{\theta} = J(\theta)^{-1} \mathcal{V}$$

What if **J(q)** is not square (and full rank) hence the inverse does not exist?

Underactuation
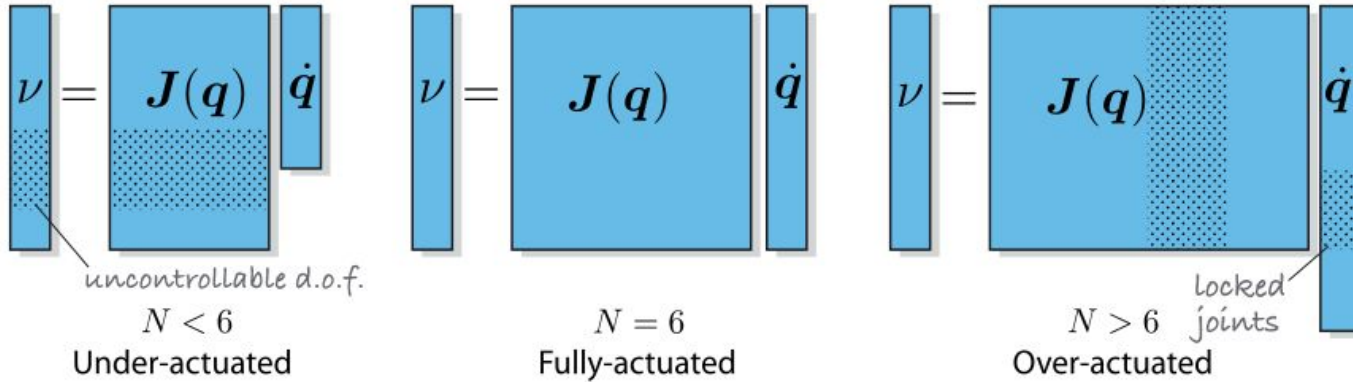
$$\nu = J(q) \dot{q}$$
uncontrollable d.o.f.
$N < 6$
Under-actuated

$$\nu = J(q) \dot{q}$$
$N = 6$
Fully-actuated

$$\nu = J(q) \dot{q}$$
locked joints
$N > 6$
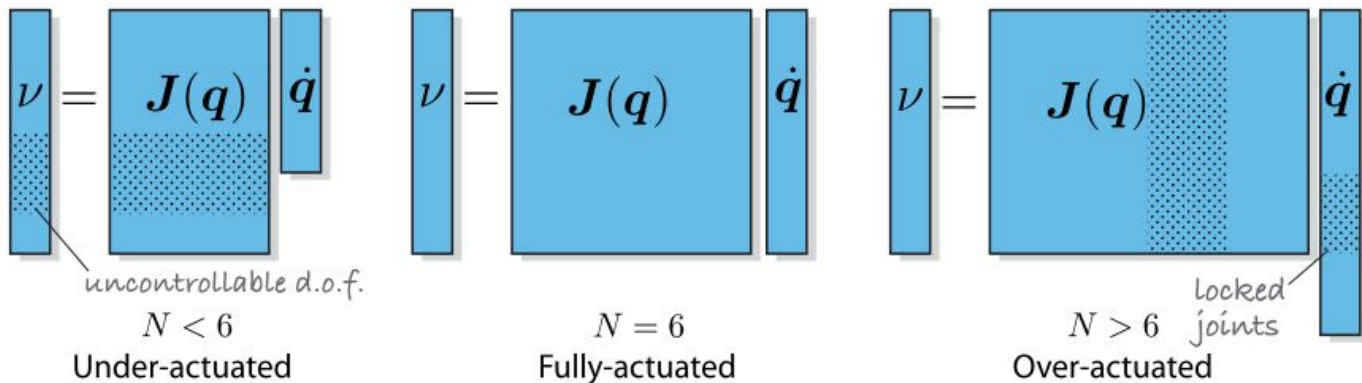Over-actuated

n=5, m = 6 (x,y,z, ϕ, θ, ψ)

$$\dot{\theta} = J(\theta)^{-1} \mathcal{V}$$

- What if **J(q)** is not square (and full rank) hence the inverse does not exist?
- Underactuated case:
  - Cannot be solved (unless we are lucky) because the system of equations is overdetermined.
  - "Quick hack": The system can be squared up by deleting some rows of v and J – accepting that some Cartesian degrees of freedom are not controllable given the low number of joints.

HeadYaw
HeadPitch
ShoulderPitch
ShoulderRoll
ElbowYaw
ElbowRoll
HipYawPitch
WristYaw
Hand
HipRoll
HipPitch
KneePitch
AnklePitch
AnkleRoll

Section 8.2 in Corke, P. I. (2013). Robotics, vision and control: fundamental algorithms in MATLAB Berlin: Springer.

34

Overactuation

n=7 (or 10), m = 6 (x,y,z, ϕ, θ, ψ)

$\nu = J(q) \dot{q}$

uncontrollable d.o.f.
$N < 6$
Under-actuated

$\nu = J(q) \dot{q}$

$N = 6$
Fully-actuated

$\nu = J(q) \dot{q}$

locked joints
$N > 6$
Over-actuated

$$\dot{\theta} = J(\theta)^{-1} \mathcal{V}$$

- What if **J(q)** is not square (and full rank) hence the inverse does not exist?
- Overractuated case:
  - The system of equations is underdetermined. Multiple solutions exist and we can find a least squares solution (later).
  - "Quick hack": Alternatively we can square up the Jacobian to make it invertible by deleting some columns – effectively locking the corresponding axes.
  - What do we do with these extra axes?

Section 8.2 in Corke, P. I. (2013). Robotics, vision and control: fundamental algorithms in MATLAB Berlin: Springer.

35

Self-motion

2.7.1 Self-Motion
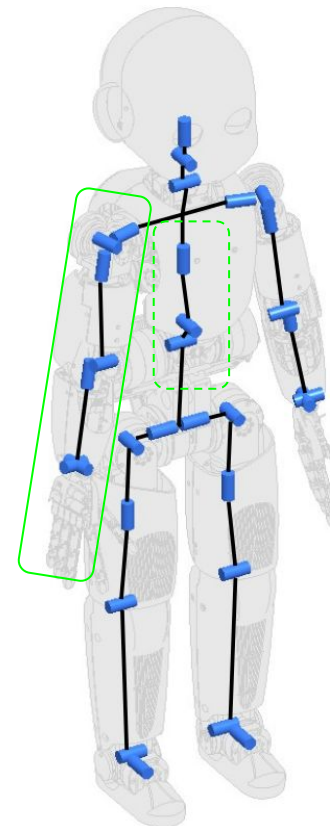
In contrast to a nonredundant limb, a kinematically redundant limb can move even when its end link is immobilized ($\mathcal{V} = \mathbf{0}$). Such motion is shown in Fig. 2.9 for the arm; the hand remains fixed w.r.t. the arm root frame while the elbow rotates around the line connecting the shoulder and wrist joints. Such type of motion is known as *self-motion*, *internal motion*, or *null motion*.

Self-motion is generated by the joint velocity obtained from the following homogeneous differential relation:

$$J(\theta)\dot{\theta} = \mathbf{0}, \quad \dot{\theta} \neq \mathbf{0}. \tag{2.28}$$

Since $n > 6$, the Jacobian is nonsquare ($6 \times n$) and the above equation is characterized as an underdetermined linear system. Hence, there is an infinite set of solutions, each nontrivial
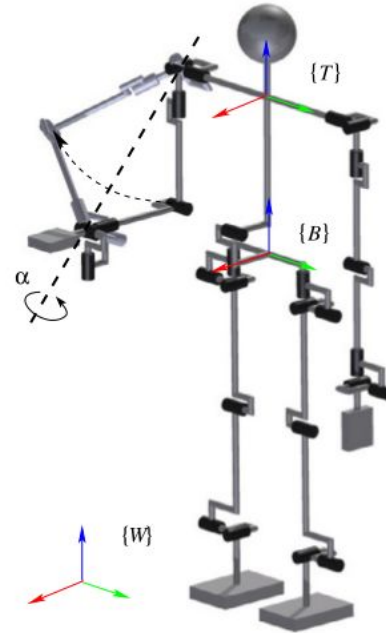

FIGURE 2.9 The self-motion of the arm is shown as a rotation of the arm plane, determined by the upper/lower arm links, around the line connecting the shoulder and wrist joints. The rotation angle α can be associated with parameter b_v in (2.35).

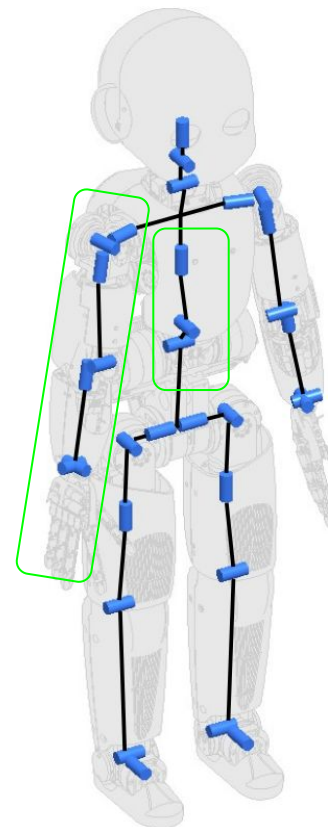Section 2.7.1 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.

Inverse differential kinematic relations

- Given the joint angles and the end-link spatial velocity, find the motion rates in the joints.
- In order to find a solution in a straightforward manner, the following two conditions have to be satisfied:
  a. the Jacobian matrix at branch configuration θ should be of full rank;
  b. the number of joints of the branch should be equal to the DoF of the end link.
- These conditions imply that the inverse of the Jacobian matrix exists.
- When the conditions are satisfied, solving $\mathcal{V}_n = J(\theta)\dot{\theta}$ the joint rates yields the following solution to the inverse kinematics problem:

$$\dot{\theta} = J(\theta)^{-1}\mathcal{V}$$

- A branch configuration yielding a full-rank Jacobian is called a *nonsingular configuration.*
- A branch with a number of joints that conforms to the second condition is called a kinematically nonredundant branch.

Section 2.4.3 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.

Inverse differential kinematic relations – challenges

- Whenever any of the above two conditions cannot be met, the inverse problem needs to be handled with care.
- Special branch configurations where the Jacobian loses rank. Such configurations are called *singular*.
  - The branch can attain a singular configuration irrespective of the number of its joints.
- Further on, when the branch comprises more joints than the DoF of its end link (n > 6), then $\mathcal{V}_n = \boldsymbol{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$ is underdetermined. This implies the existence of an infinite set of inverse kinematics solutions for the joint rates. In this case, the branch is referred to as a *kinematically redundant branch*.

Section 2.4.3 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.

Generalized inverse

- We are looking for a matrix **J#** such that: $\dot{\theta} = J^{\#}\nu$
- Two cases:
  - Underactuated manipulator (~ overdetermined system of equations): Find $\dot{\theta}$

    such that $J\dot{\theta} - \nu$ is minimized.

  - Redundant manipulator (~ underdetermined system of equations): Find $\dot{\theta}$

    solving $\nu = J\dot{\theta}$ optimizing some additional property.

Forward, inverse, and differential kinematics – recap

	Mapping type	linear / nonlinear
$\mathbf{x} = \mathbf{f}(\mathbf{q})$	Many-to-one (for n>= m)	nonlinear
$\mathbf{q} = \mathbf{f}^{-1}(\mathbf{x})$	One-to-many (for n>1)	nonlinear
$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$		
$\dot{\mathbf{q}} = \mathbf{J}^{-1}\dot{\mathbf{x}}$		

$p_x = \cos\theta_1\,(a_3\cos(\theta_2 + \theta_3) + a_2\cos\theta_2)$
$p_y = \sin\theta_1\,(a_3\cos(\theta_2 + \theta_3) + a_2\cos\theta_2)$
$p_z = a_3\sin(\theta_2 + \theta_3) + a_2\sin\theta_2 + a_1$
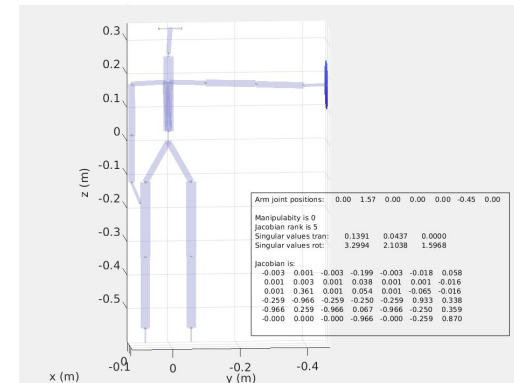
https://www.slideserve.com/antonia/inverting-the-jacobian-and-manipulability

Forward, inverse, and differential kinematics – recap

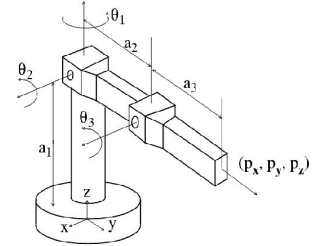|	Mapping type	linear / nonlinear
$\mathbf{x} = \mathbf{f}(\mathbf{q})$	Many-to-one (for n>= m)	nonlinear
$\mathbf{q} = \mathbf{f}^{-1}(\mathbf{x})$	One-to-many (for n>1)	nonlinear
$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$	Many-to-one (for n>= m)	linear
$\dot{\mathbf{q}} = \mathbf{J}^{-1}\dot{\mathbf{x}}$	One-to-many (for n>1)	linear

$$p_x = \cos\theta_1 \, (a_3 \cos(\theta_2 + \theta_3) + a_2 \cos\theta_2)$$
$$p_y = \sin\theta_1 \, (a_3 \cos(\theta_2 + \theta_3) + a_2 \cos\theta_2)$$
$$p_z = a_3 \sin(\theta_2 + \theta_3) + a_2 \sin\theta_2 + a_1$$

41

Using differential kinematics for IK

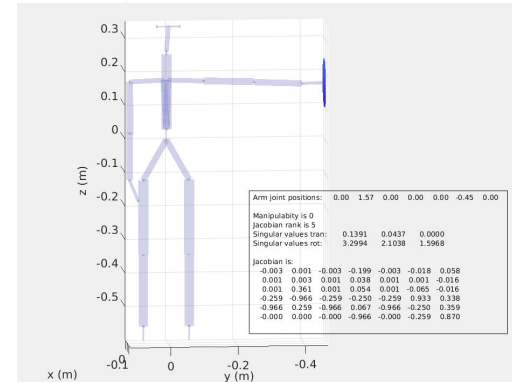- They are numerical, iterative methods.
- Methods relying on some form of Jacobian inverse.
  - Jacobian transpose
  - Jacobian pseudoinverse
    - Used by Orocos Kinematics and Dynamics Library (KDL) - used in ROS.
  - Damped least squares
- Methods using optimization to find solutions to forward differential kinematics.
- More details in
  - Haviland, J., & Corke, P. (2022). Manipulator Differential Kinematics
    - Part I: Kinematics, Velocity, and Applications. https://arxiv.org/pdf/2207.01796.pdf
    - Part II: Acceleration and Advanced Applications. https://arxiv.org/pdf/2207.01794.pdf
  - Buss, S. R. (2004). Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods. IEEE Journal of Robotics and Automation, 17(1-19), 16.
  - 2.7.2 - 2.7.5 in Nenchev et al. (2018)
  - Jacobian transpose - duality of kinematics and statics - 8.4 in Corke, P. I. (2013). Robotics, vision and control: fundamental algorithms in MATLAB Berlin: Springer.
  - https://github.com/vvv-school/vvv18/blob/master/material/kinematics/kinematics.pdf
  - https://www.slideserve.com/antonia/inverting-the-jacobian-and-manipulability

Differential kinematics for IK – Jacobian "inversion"

Jacobian transpose

We aim to minimize the Cartesian error:
$$g = \frac{1}{2}\|\mathbf{e}\|^2 = \frac{1}{2}\|\mathbf{x}_d - \mathbf{f}(\mathbf{q})\|^2$$

Compute the gradient:
$$\nabla_{\mathbf{q}}g = \frac{1}{2}\nabla_{\mathbf{q}}\left\langle(\mathbf{x}_d - \mathbf{f}),(\mathbf{x}_d - \mathbf{f})\right\rangle = \frac{1}{2}\cdot\left(-2\left\langle\nabla_{\mathbf{q}}\mathbf{f},(\mathbf{x}_d - \mathbf{f})\right\rangle\right) =$$
$$= -\left\langle\mathbf{J},(\mathbf{x}_d - \mathbf{f})\right\rangle = -\mathbf{J}^T(\mathbf{x}_d - \mathbf{f}) = -\mathbf{J}^T\mathbf{e}$$

Gradient descent method for system of nonlinear equations:
$$\dot{\mathbf{q}} = \mathbf{K}\cdot(-\nabla_{\mathbf{q}}g) = \mathbf{J}^T\mathbf{K}\mathbf{e}$$

we only employ *direct kinematics functions*!

Jacobian transpose from statics

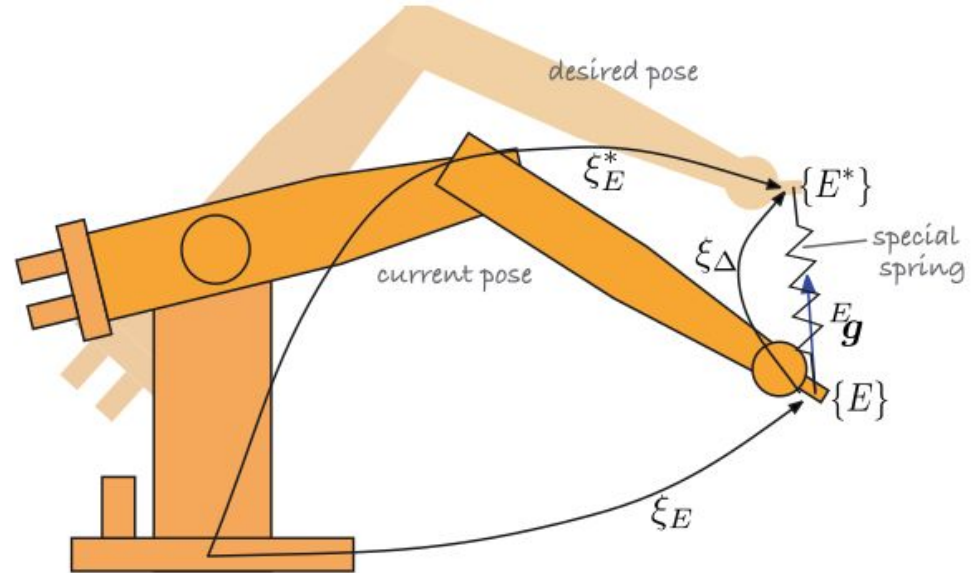**Fig. 8.7.**
Schematic of the numerical inverse kinematic approach, showing the current ξ_E and the desired ξ_E^* manipulator pose

8.4 in Corke, P. I. (2013). Robotics, vision and control: fundamental algorithms in MATLAB Berlin: Springer.

Jacobian pseudoinverse

redundant chain $n > m$

Reformulate the problem as a *linear constrained optimization*

$$\dot{\mathbf{q}}^* = \arg \min_{\dot{\mathbf{q}}} \left(\frac{1}{2} \dot{\mathbf{q}}^T \mathbf{W} \dot{\mathbf{q}} \right)$$

$$\text{s.t.} \ \ \dot{\mathbf{x}} = \mathbf{J} \dot{\mathbf{q}}$$

Recruit Lagrangian multipliers:

$$\dot{\mathbf{q}}^* = \arg \min_{\dot{\mathbf{q}}, \lambda} \left(\frac{1}{2} \dot{\mathbf{q}}^T \mathbf{W} \dot{\mathbf{q}} + \lambda^T \left(\dot{\mathbf{x}} - \mathbf{J} \dot{\mathbf{q}} \right) \right)$$

$$\dot{\mathbf{q}}^* = \mathbf{W}^{-1} \mathbf{J}^T \left(\mathbf{J} \mathbf{W}^{-1} \mathbf{J}^T \right)^{-1} \dot{\mathbf{x}}$$

$$\begin{cases} \mathbf{W} = \mathbf{I}_n \\ \mathbf{J}^\dagger = \mathbf{J}^T \left(\mathbf{J} \mathbf{J}^T \right)^{-1} \ \Rightarrow \ \dot{\mathbf{q}} = \mathbf{J}^\dagger \dot{\mathbf{x}} \\ \mathbf{J} \mathbf{J}^\dagger = \mathbf{I}_n \end{cases}$$

46

<u>Psuedoinverse definition:</u> (underconstrained)

Given a desired twist, $\dot{x}_d$, find a vector of
joint velocities, $\dot{q}$, that satisfies $\dot{x}_d = J\dot{q}$
while minimizing $f(\dot{q}) = \dot{q}^T \dot{q}$

Minimize joint velocities

Minimize $f(z)$ subject to $g(z) = 0$:

Use **lagrange multiplier method**: $\nabla_z f(z) = \lambda \nabla_z g(z)$

This condition must be met when $f(z)$ is at a minimum
subject to $g(z) = 0$

Howie Choset: https://www.slideserve.com/antonia/inverting-the-jacobian-and-manipulability   47
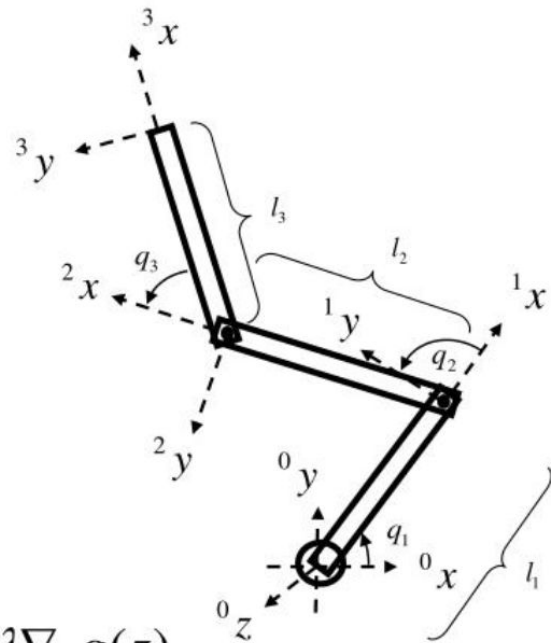
$$\nabla_z f(z) = \lambda \nabla_z g(z)$$

$$f(\dot{q}) = \tfrac{1}{2} \dot{q}^T \dot{q} \quad \longleftarrow \quad \text{Minimize}$$

$$g(\dot{q}) = J\dot{q} - \dot{x} = 0 \quad \longleftarrow \quad \text{Subject to}$$

$$\nabla_{\dot{q}} f(\dot{q}) = \dot{q}^T$$

$$\nabla_{\dot{q}} g(\dot{q}) = J$$

$$\dot{q}^T = \lambda^T J$$

$$\dot{q} = J^T \lambda$$

$$\dot{q} = J^T \lambda$$

$$J\dot{q} = (JJ^T)\lambda$$

$$\lambda = (JJ^T)^{-1} J\dot{q}$$ ⟵ I won't say why, but if J is full rank, then JJ^T is invertible

$$\lambda = (JJ^T)^{-1} \dot{x}$$

$$\dot{q} = J^T \lambda$$

$$\dot{q} = J^T (JJ^T)^{-1} \dot{x}$$

So, the pseudoinverse calculates the vector of joint velocities that satisfies $\dot{x}_d = J\dot{q}$ while minimizing the squared magnitude of joint velocity ($\dot{q}^T \dot{q}$).

$$J^\# = J^T (JJ^T)^{-1}$$

$$\dot{q} = J^\# \dot{x}$$ ⟵

- Therefore, the pseudoinverse calculates the *least-squares* solution.

Calculating the pseudoinverse

The pseudoinverse can be calculated using two different equations depending upon the number of rows and columns:

$$J^{\#} = J^T \left(J J^T \right)^{-1}$$ Underconstrained case (if there are more columns than rows ($m<n$))

$$J^{\#} = \left(J^T J \right)^{-1} J^T$$ Overconstrained case (if there are more rows than columns ($n<m$))

$$J^{\#} = J^{-1}$$ If there are an equal number of rows and columns ($n=m$)

These equations can only be used if the Jacobian is full rank; otherwise, use singular value decomposition (SVD):

Calculating the pseudoinverse using SVD

Singular value decomposition decomposes a matrix as follows:

$$J = U \Sigma V^T$$

$m \times m \quad m \times n \quad n \times n$
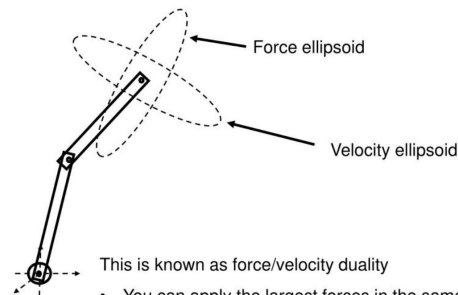
Σ is a diagonal matrix of singular values:

$$J = U \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_n & 0 & 0 \end{bmatrix} V^T$$

$$J^{\#} = V \Sigma^{-1} U^T$$

$$J^{\#} = V \begin{bmatrix} \frac{1}{\sigma_1} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma_2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sigma_3} & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sigma_n} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} U^T$$

~ manipulability (velocity) ellipsoid

Velocity and force manipulability are orthogonal!

Force ellipsoid

Velocity ellipsoid

This is known as force/velocity duality
- You can apply the largest forces in the same directions that your max velocity is smallest
- Your max velocity is greatest in the directions where you can only apply the smallest forces

~ force ellipsoid

Howie Choset: https://www.slideserve.com/antonia/inverting-the-jacobian-and-manipulability

Properties of the pseudoinverse

Moore-Penrose conditions:

1. $J^{\#}JJ^{\#} = J^{\#}$
2. $JJ^{\#}J = J$
3. $\left(JJ^{\#}\right)^{T} = JJ^{\#}$
4. $\left(J^{\#}J\right)^{T} = J^{\#}J$

Generalized inverse: satisfies condition 1

Reflexive generalized inverse: satisfies conditions 1 and 2

Pseudoinverse: satisfies all four conditions

Other useful properties of the pseudoinverse:

$$\left(J^{\#}\right)^{\#} = J$$

$$\left(J^{\#}\right)^{T} = \left(J^{T}\right)^{\#}$$

Howie Choset: https://www.slideserve.com/antonia/inverting-the-jacobian-and-manipulability

Secondary task

redundant chain $n > m$

Reformulate the problem as a *linear constrained optimization*

$$\dot{\mathbf{q}}^* = \arg \min_{\dot{\mathbf{q}}} \left(\frac{1}{2} (\dot{\mathbf{q}} - \dot{\mathbf{q}}_0)^T (\dot{\mathbf{q}} - \dot{\mathbf{q}}_0) \right)$$

$$\text{s.t. } \dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$$

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\dot{\mathbf{x}} + \left(\mathbf{I}_n - \mathbf{J}^{\dagger}\mathbf{J} \right) \dot{\mathbf{q}}_0$$

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_0 + \mathbf{J}^{\dagger} \left(\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}}_0 \right)$$

$$\left(\mathbf{I}_n - \mathbf{J}^{\dagger}\mathbf{J} \right)$$ Null-space projection operator allows for *internal motions*

How to pick up $\dot{\mathbf{q}}_0$:

$$\dot{\mathbf{q}}_0 = k_0 \left(\frac{\partial w(\mathbf{q})}{\partial \mathbf{q}} \right)^T$$
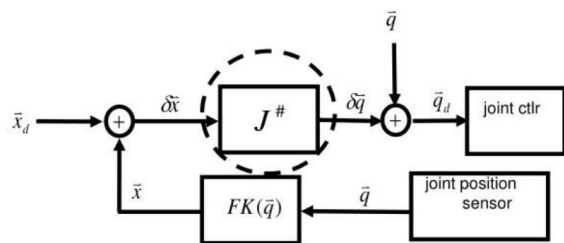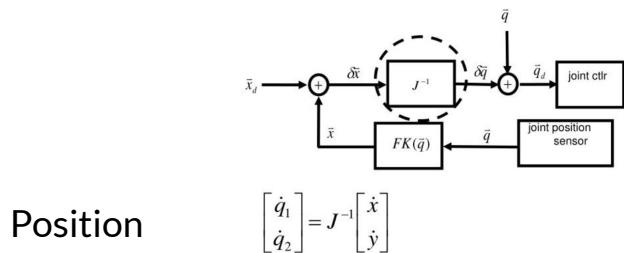
Improve manipulability

$$w(\mathbf{q}) = \sqrt{\det \left(\mathbf{J}(\mathbf{q}) \mathbf{J}^T(\mathbf{q}) \right)}$$

Joint limits avoidance

$$w(\mathbf{q}) = -\frac{1}{2n} \sum_{i=1}^{n} \left(\frac{q_i - \overline{q}_i}{q_{iM} - q_{im}} \right)^2$$

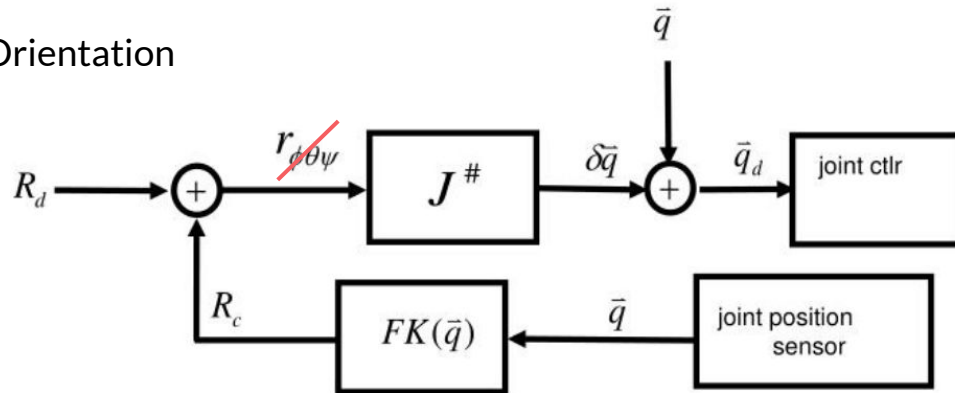# Using J# for Cartesian control

Orientation

Position

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = J^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

Remember that in general: $J_\omega \neq \dfrac{\partial r_{\phi\theta\psi}}{\partial q}$

Use analytical Jacobian or axis-angle representation and Rodrigues formula. (mechanical vs. representational singularities...)

See slides 16-20 in
https://www.slideserve.com/antonia/inverting-the-jacobian-and-manipulability

Procedure for controlling position:

1. Calculate position error: x_{err}
2. Multiply by a scaling factor: $\delta x_{err} = \alpha x_{err}$
3. Multiply by the velocity Jacobian pseudoinverse: $\dot{q} = J_v^{\#} \alpha x_{err}$

Howie Choset: https://www.slideserve.com/antonia/inverting-the-jacobian-and-manipulability

Jacobian transpose vs. pseudoinverse

- Which is more direct? Jacobian pseudoinverse or transpose?

$$\dot{q} = J^T \xi \qquad \text{or} \qquad \dot{q} = J^{\#} \xi$$

They do different things:

- Transpose: move toward a reference pose as quickly as possible
  - One dimensional goal (squared distance meteric)
- Pseudoinverse: move along a least squares reference twist trajectory
  - Six dimensional goal (or whatever the dimension of the relevant twist is)

Howie Choset: https://www.slideserve.com/antonia/inverting-the-jacobian-and-manipulability

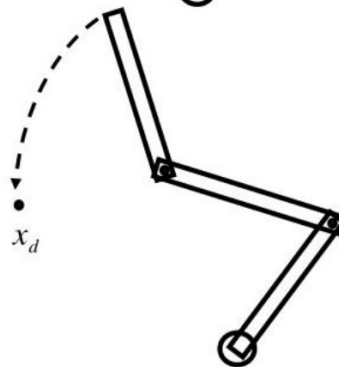# Jacobian transpose vs. pseudoinverse

The pseudoinverse moves the end effector in a straight line path toward the goal pose using the least squared joint velocities.

- The goal is specified in terms of the reference twist

- Manipulator follows a straight line path in Cartesian space
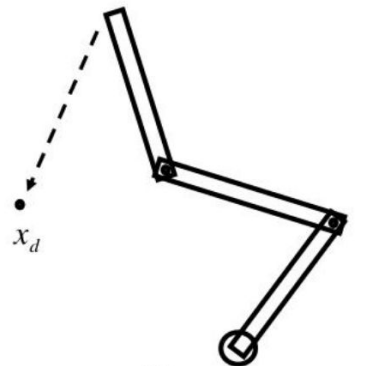
The transpose moves the end effector toward the goal position

- In general, not a straight line path *in Cartesian space*

- Instead, the transpose follows the gradient in *joint space*

x_d

x_d

Damped Least Squares

To deal with kinematic singularities

$$\dot{\mathbf{q}}^* = \arg\min_{\dot{\mathbf{q}}} \left(\frac{1}{2}(\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}})^T (\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}}) + \frac{1}{2}k^2\dot{\mathbf{q}}^T\dot{\mathbf{q}} \right)$$

$$\mathbf{J}^* = \mathbf{J}^T \left(\mathbf{J}\mathbf{J}^T + k^2\mathbf{I}_n \right)^{-1}$$

Levenberg-Marquardt method
for system of nonlinear equations

$$\dot{\mathbf{q}} = \mathbf{J}^*\dot{\mathbf{x}}$$

k establishes a synergy between:
- Jacobian (Pseudo-)inverse: $k = 0$
- Jacobian Transpose: $k \gg \max\{|\sigma_i|\}$

Differential kinematics for IK – Jacobian "inversion" overview

Iterative Methods

Jacobian Transpose

$$\dot{\mathbf{q}} = \mathbf{J}^T \mathbf{K} \mathbf{e}, \quad \mathbf{e} = \mathbf{x}_d - \mathbf{x}_e$$

Jacobian Pseudoinverse

$$\dot{\mathbf{q}} = \mathbf{J}^\dagger \mathbf{K} \mathbf{e} + \left(\mathbf{I} - \mathbf{J}^\dagger \mathbf{J} \right) \dot{\mathbf{q}}_0, \quad \mathbf{J}^\dagger = \mathbf{J}^T \left(\mathbf{J}\mathbf{J}^T \right)^{-1}$$

Damped Least Squares

$$\dot{\mathbf{q}} = \mathbf{J}^* \mathbf{K} \mathbf{e}, \quad \mathbf{J}^* = \mathbf{J}^T \left(\mathbf{J}\mathbf{J}^T + k^2 \mathbf{I} \right)^{-1}$$

- More details in
    - Buss, S. R. (2004). Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods. IEEE Journal of Robotics and Automation, 17(1-19), 16.
    - 2.7.2 - 2.7.5 in Nenchev et al. (2018); Redundancy resolution 2.7.4-2.7.5
    - Jacobian transpose - duality of kinematics and statics - 8.4 in Corke, P. I. (2013). Robotics, vision and control: fundamental algorithms in MATLAB Berlin: Springer.
    - https://github.com/vvv-school/vvv18/blob/master/material/kinematics/kinematics.pdf
    - Howie Choset: https://www.slideserve.com/antonia/inverting-the-jacobian-and-manipulability

IK is an active research topic

- Orocos Kinematics and Dynamics Library (KDL) - used in ROS / MoveIt!
  - Uses joint-limit constrained pseudoinverse Jacobian solver

Despite its popularity, KDL's IK implementation[3] exhibits numerous false-negative failures on a variety of humanoid and mobile manipulation platforms. In particular, KDL's IK implementation has the following issues:

1) frequent convergence failures for robots with joint limits,
2) no actions taken when the search becomes "stuck" in local minima,
3) inadequate support for Cartesian pose tolerances,
4) no utilization of tolerances in the IK solver itself.

Beeson, P., & Ames, B. (2015, November). TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)* (pp. 928-935). IEEE.

The current state-of-the-art and defacto-usage of numerical inverse kinematics fall into variations of these aforementioned algorithms. The open-source *Orocos Kinematics and Dynamics Library* (KDL) [17] is arguably the most popular generic inverse kinematics solver [7], and features various solvers based on a Levenberg–Marquard optimization approach, as well as a Newton-Raphson pseudoinverse solver. The KDL library is used in the ROS libraries, along with another solver called Trac-IK that improves reliability by simultaneously running both an Newton-Raphson solver and a BFGS solver, and stopping when either of the two reaches a successful solution [7]. Mathworks' popular *MATLAB Robotics Toolbox* features two solvers—one based on the Levenberg–Marquard algorithm (implemented as in [12]), and another based on the BFGS algorithm (implemented as in [1]).

Lloyd, S., Irani, R. A., & Ahmadi, M. (2022). Fast and Robust Inverse Kinematics of Serial Robots Using Halley's Method. IEEE Transactions on Robotics, 38(5), 2768-2780.

Differential kinematics for IK – "forward formulation" – optimization

Reactive velocity controller with whole–body obstacle avoidance
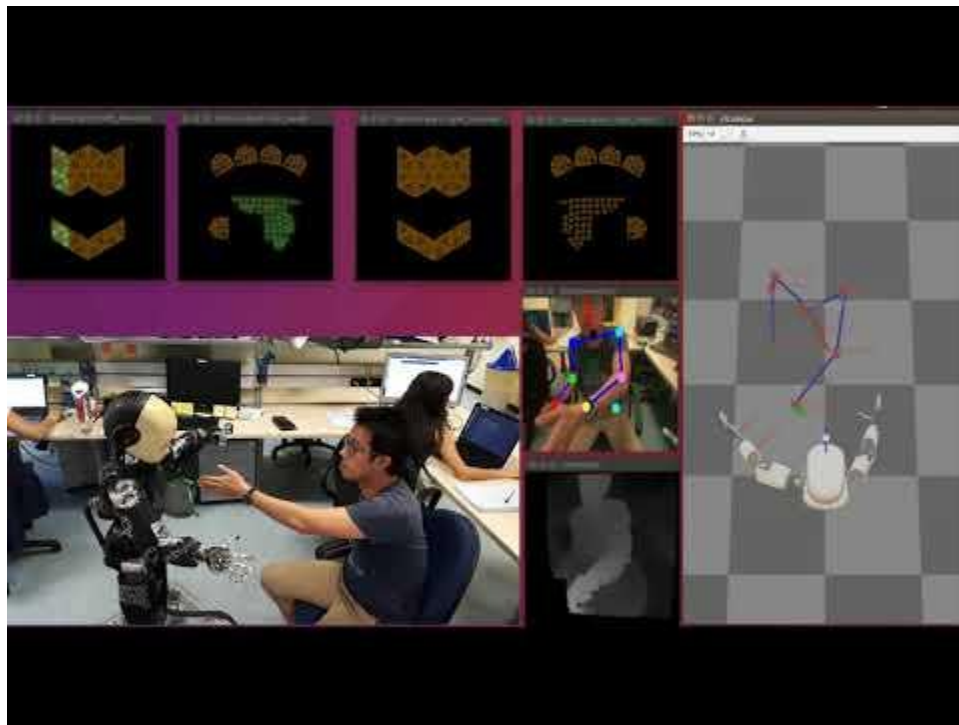
Nguyen, P. D.; Hoffmann, M.; Roncone, A.; Pattacini, U. & Metta, G. (2018), Compact real-time avoidance on a humanoid robot for human-robot interaction, *in* 'HRI '18: 2018 ACM/IEEE International Conference on Human-Robot Interaction', ACM, New York, NY, USA, pp. 416-424.[ACM digital library][arxiv]
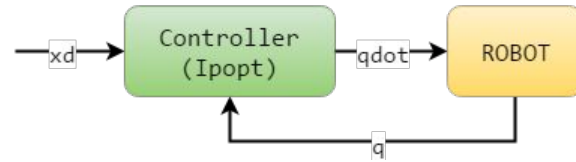
New reaching controller

- Previously: iCub inverse kinematics solver and Cartesian controller decoupled (Pattacini et al. 2010)

- Now: single velocity solver + controller

$$\dot{\mathbf{q}}_d = \arg \min_{\dot{\mathbf{q}} \in \mathbb{R}^n} \left(\left\| \overline{\mathbf{x}}_{EEd} - \left(\overline{\mathbf{x}}_{EE} + T_S \mathbf{J}(\overline{\mathbf{q}}) \dot{\mathbf{q}} \right) \right\|^2 \right)$$

$$\text{s.t.} \begin{cases} \mathbf{q}_L < \overline{\mathbf{q}} + T_S \dot{\mathbf{q}} < \mathbf{q}_U \\ \dot{\mathbf{q}}_L < \dot{\mathbf{q}} < \dot{\mathbf{q}}_U \\ \text{other constraints...} \end{cases}$$

obstacles $\longrightarrow$

$$\begin{cases} \mathbf{s} = -\mathbf{J}_C^T \mathbf{n}_C \cdot V_C \cdot a_{PPS} \\ \dot{q}_{L,i} = \max\left\{V_{L,i}, s_i\right\} \quad s_i > 0 \\ \dot{q}_{U,i} = \min\left\{V_{U,i}, s_i\right\} \quad s_i < 0 \end{cases}$$

- **Local constrained minimization** @ each instant $\bar{t}$ of Cartesian distances with one-step look-ahead $\bar{t} + T_s$
- Search space: **joint velocities**
- **Fast convergence**, can run @ $T_s \leq 10$ ms
- Use of consolidated optimization library: **Ipopt**
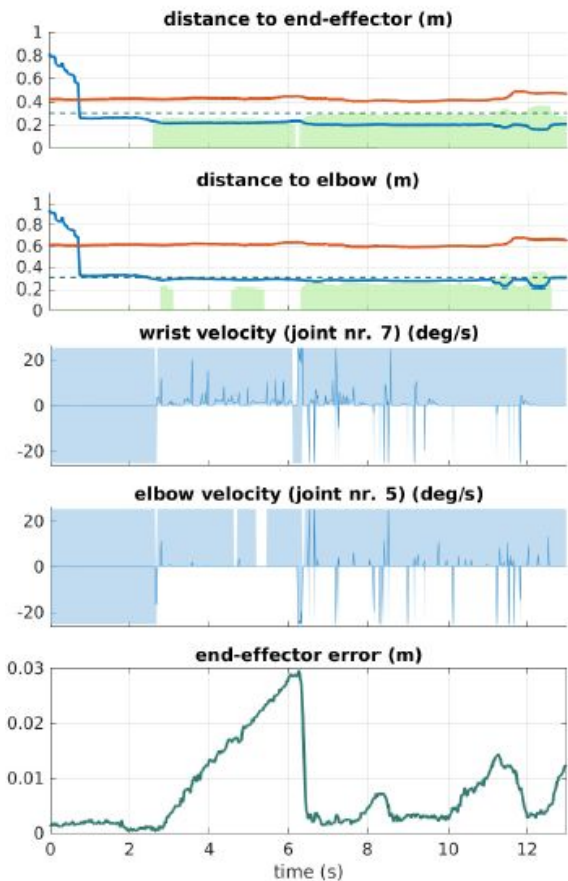
Reaching Controller: Obstacles

- Mapping multiple Cartesian obstacles / repulsive vectors onto joint velocity limits to bring about avoidance

$$
\begin{cases}
\mathbf{s} = -\mathbf{J}_C^T \mathbf{n}_C \cdot V_C \cdot a_{PPS} \\
\dot{q}_{L,i} = \max\left\{V_{L,i}, s_i\right\} \quad s_i > 0 \\
\dot{q}_{U,i} = \min\left\{V_{U,i}, s_i\right\} \quad s_i < 0
\end{cases}
$$

- Control points (C) – loci of PPS activations
- V_C – gain factor for avoidance
- a_{PPS} – PPS activation
- V_L, V_U – bounding values of joint velocity
- s_j - degree of influence of the Cartesian constraint on the j-th joint
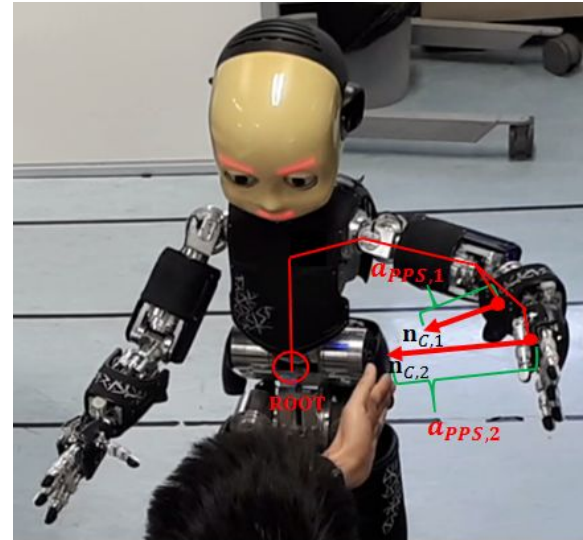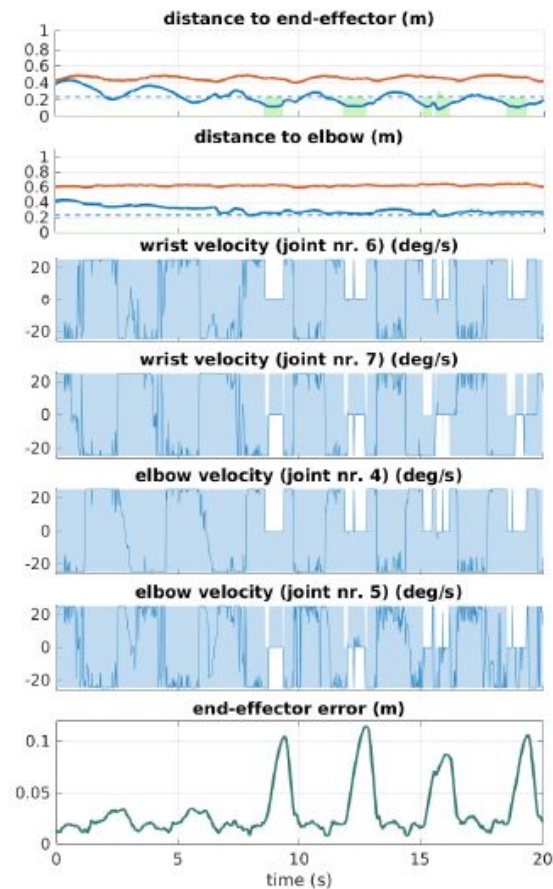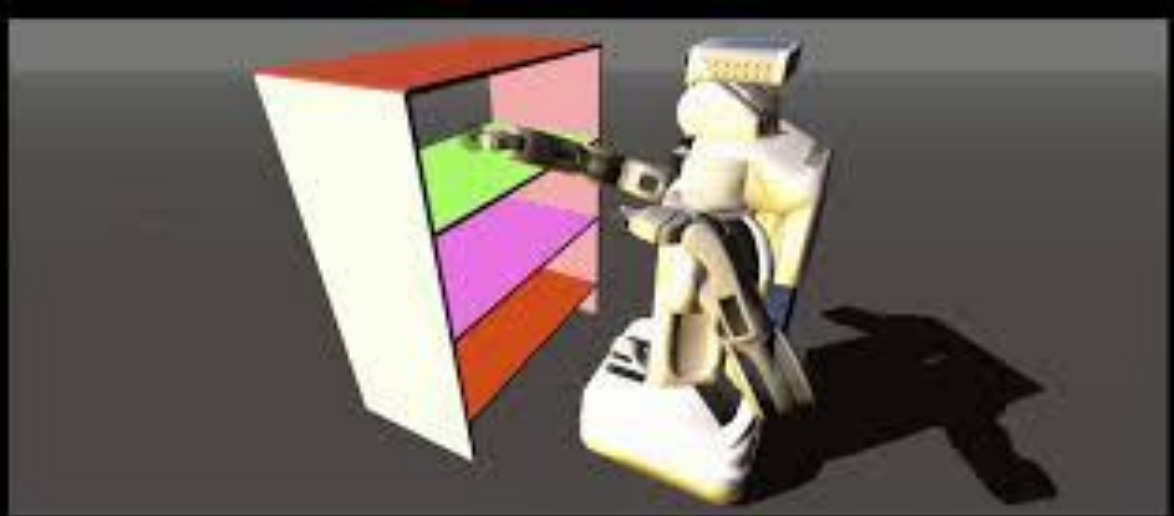
static target

& simultaneous
obstacle avoidance

moving target

NEO – quadratic programming

NEO is a purely reactive motion controller which can dodge stationary and non-stationary obstacles while respecting robot joint limits and maximising the robot's manipulability. One use case for this controller is reactive closed-loop pick and place tasks on a bookshelf.

A. Incorporating the Differential Kinematics Into a QP

The first-order differential kinematics for a serial-link manipulator are described by

$$\boldsymbol{\nu}(t) = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}(t) \qquad (2)$$

where $\boldsymbol{\nu}(t) = \begin{pmatrix} v_x & v_y & v_z & \omega_x & \omega_y & \omega_z \end{pmatrix}^\top \in \mathbb{R}^6$ is the end-effector spatial velocity and $\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{6 \times n}$ is the manipulator Jacobian which relates $\boldsymbol{\nu}$ to the joint velocities $\dot{\boldsymbol{q}}(t)$, and n is the number of joints in the robot.

We can incorporate (2) into a quadratic program as

$$\min_{\dot{\boldsymbol{q}}} \quad f_o(\dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^\top \boldsymbol{I} \dot{\boldsymbol{q}},$$

$$\text{subject to} \quad \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{\nu},$$

$$\dot{\boldsymbol{q}}^- \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}^+ \qquad (3)$$

where $\boldsymbol{I}$ is an $n \times n$ identity matrix, $\dot{\boldsymbol{q}}^-$ and $\dot{\boldsymbol{q}}^+$ represent the upper and lower joint-velocity limits, and no inequality constraints need to be defined. If the robot has more degrees-of-freedom than necessary to reach its entire task space, the QP in (3) will achieve the desired end-effector velocity with the minimum joint-velocity norm. However, there are other things we can optimise for, such as manipulability.

Haviland, J., & Corke, P. (2021). NEO: A novel expeditious optimisation algorithm for reactive motion control of manipulators. *IEEE Robotics and Automation Letters*, *6*(2), 1043-1050.

iCub playing "Bubbles"

Rozlivek, Roncone, Pattacini & Hoffmann (2023) -
in preparation
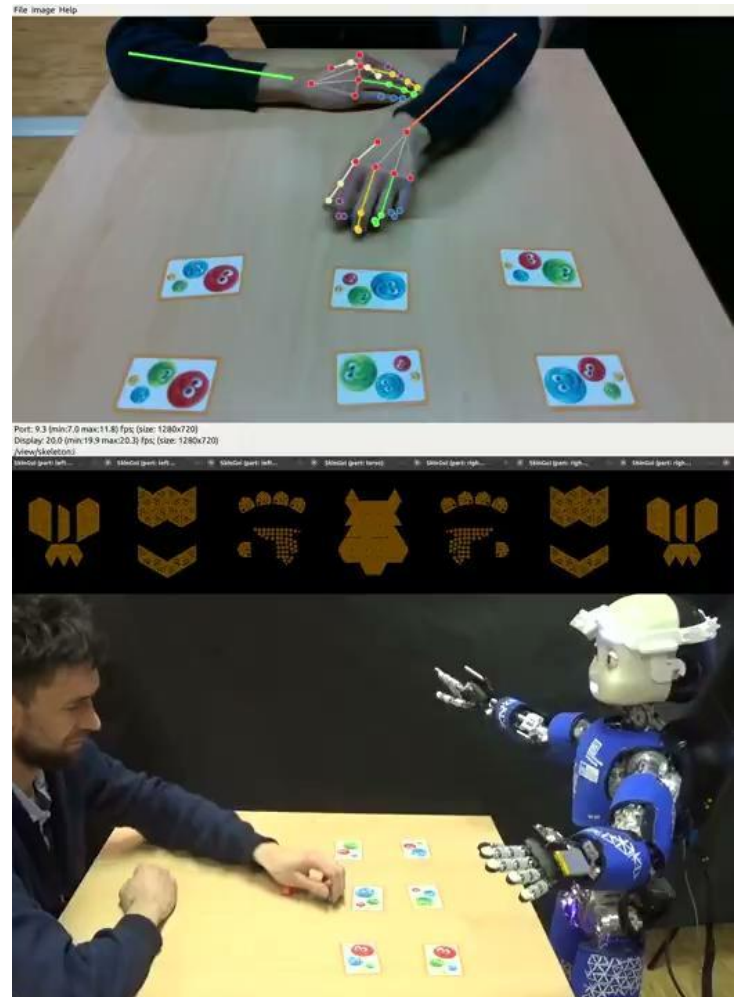
Inverse kinematics for humanoids

(a bit speculative)

There's no free lunch…

Method	General solution (not robot specific)	Takes care of motion control (trajectories)	Possibility of including additional constraints	Problems with singularities	Problems with local minima	Runtime	Examples
Analytical (closed-form)	😞	😞	😑	😑	😌	😄	
Iterative using some form of Jacobian inverse	😌	😌	😞	😞 (not DLS)	😑	😌	KDL / Orocos (ROS)
Optimization using forward kinematics	😌	😌	😑	😌	😑	😌	Velocity space - Neo (Haviland & Corke); position space - iCub IK solver

Gaze control

gazing with saccades
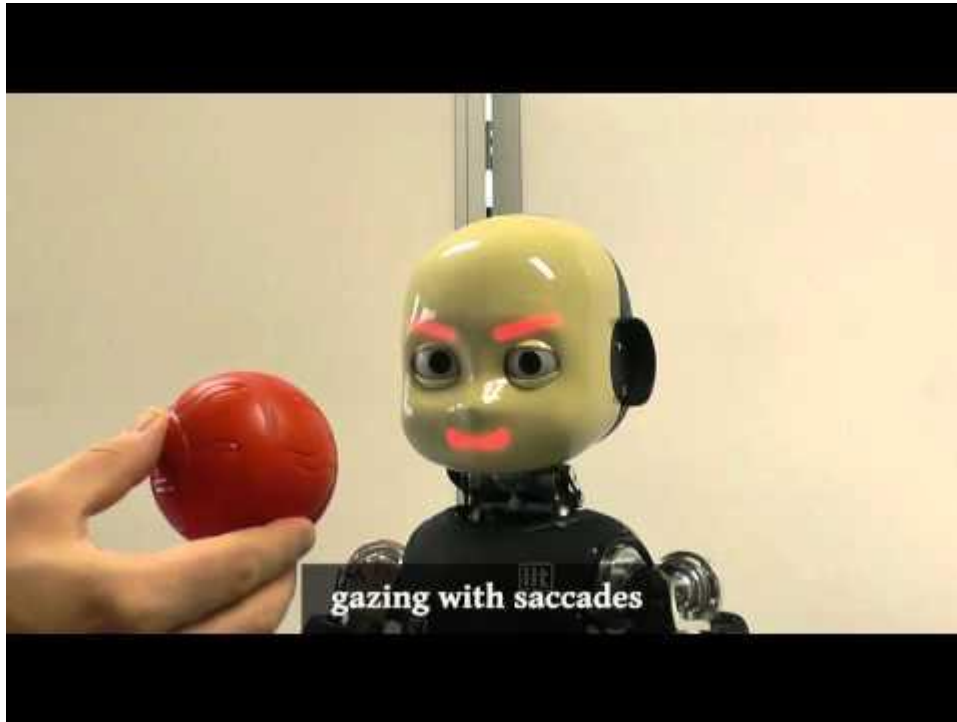
What did you see in the video?

Roncone, A., Pattacini, U., Metta, G., & Natale, L. (2016, June). A Cartesian 6-DoF Gaze Controller for Humanoid Robots. In *Robotics: science and systems* (Vol. 2016).
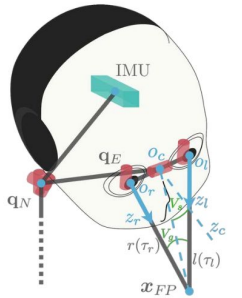
Gaze control – functions

- fixation point in image plane or 3D space
- switching attention
- tracking / smooth pursuit
- image stabilization
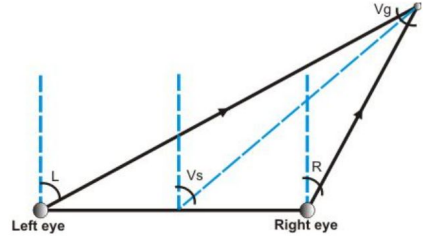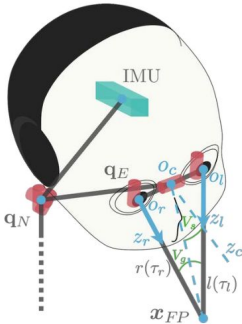- human-robot interaction - communication - e.g. joint attention…
- …

Gaze control as inverse kinematics

Joint #	Part	Joint Name	Range	Unit
0	Neck	**Pitch**	+/−	[deg]
1	Neck	**Roll**	+/−	[deg]
2	Neck	**Yaw**	+/−	[deg]
3	Eyes	**Tilt**	+/−	[deg]
4	Eyes	**Version**	+/−	[deg]
5	Eyes	**Vergence**	≥ 0	[deg]

- ## What kind of inverse kinematics problem is it?
  - ### How many DoF in joint space?
  - ### How many DoF for the task?

- ## What can the redundancy be used for?
  - vestibulo-ocular reflex (VOR)
  - saccadic behavior
  - gaze/image stabilization
  - [All three used in: Roncone, A., Pattacini, U., Metta, G., & Natale, L. (2016, June). A Cartesian 6-DoF Gaze Controller for Humanoid Robots. In Robotics: science and systems.]

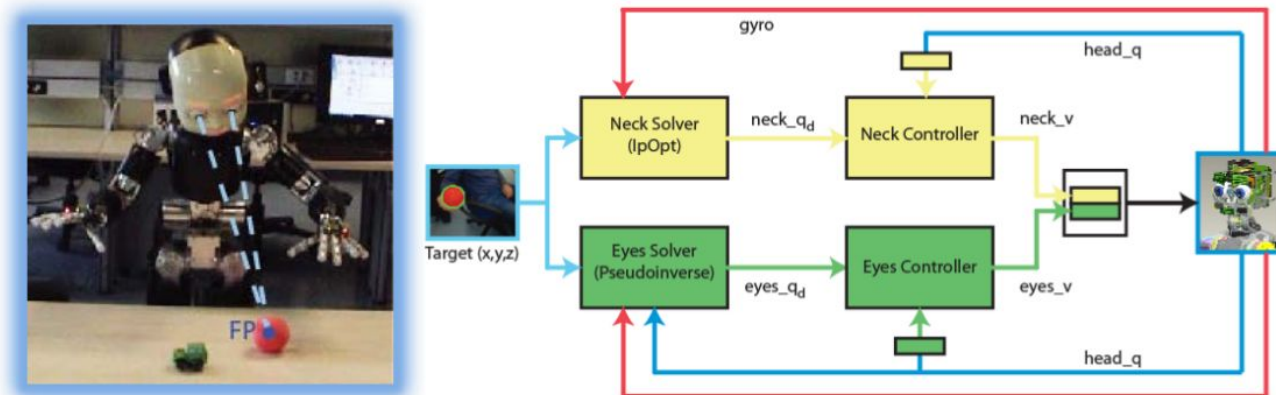$$\begin{cases} V_g = L - R \\ V_s \approx (L + R)/2 \end{cases}$$

https://github.com/vvv-school/vvv18/blob/master/material/kinematics/kinematics.pdf, courtesy Ugo Pattacini

Gaze/image stabilization
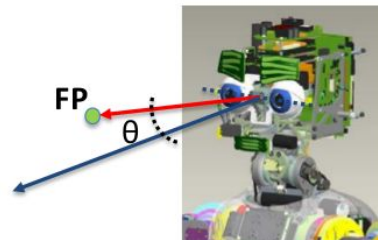

https://youtu.be/_dPlkFPowCc?t=35
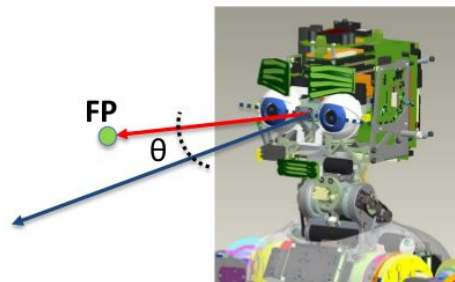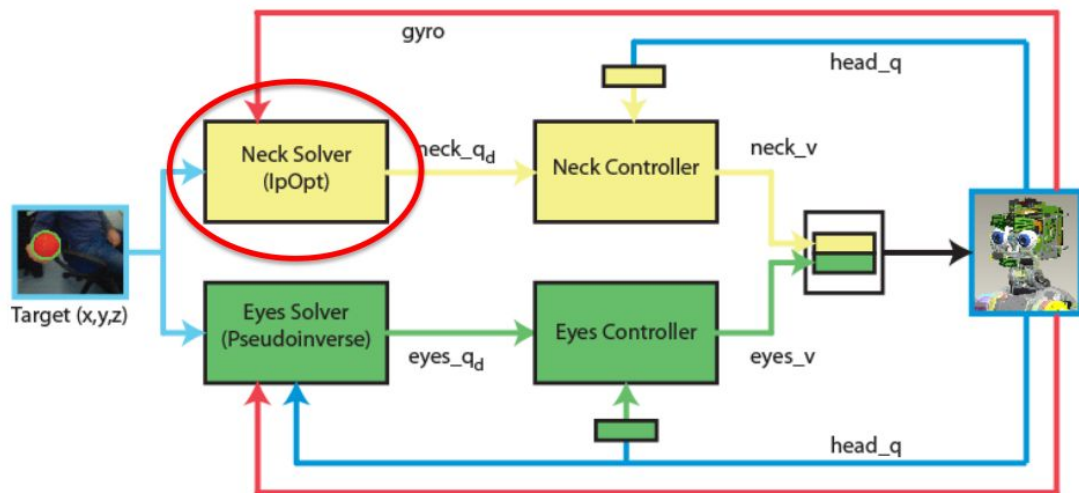
iCub gaze controller

Yet another Cartesian Controller: **reuse ideas** ...

Then, apply easy transformations from Cartesian to ...

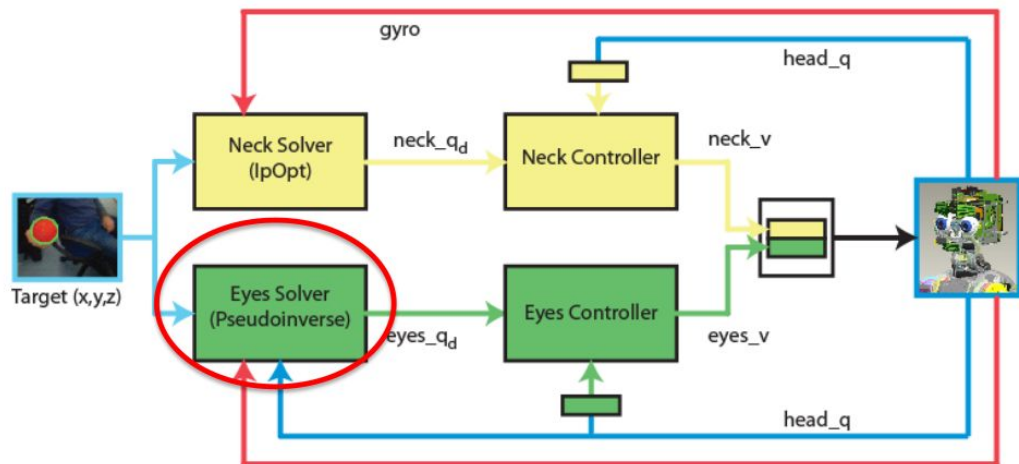1. Egocentric angular space
2. Image planes (mono and stereo)

iCub gaze controller

$$q^*_{\text{neck}} = \arg \min_{q_{\text{neck}} \in \mathbb{R}^3} \left\| q_{\text{rest}} - q_{\text{neck}} \right\|^2$$

$$\text{s.t.} \begin{cases} \cos\left(\theta\left(q_{\text{neck}}\right)\right) > 1 - \varepsilon \\ q_{\text{neck}_L} < q_{\text{neck}} < q_{\text{neck}_U} \end{cases}$$

iCub gaze controller

$$q_{\text{eyes}}^{*} = \arg \min_{q_{\text{eyes}} \in \mathbb{R}^3} \left\| FP_d - K_{FP}\left(q_{\text{eyes}}\right) \right\|^2$$
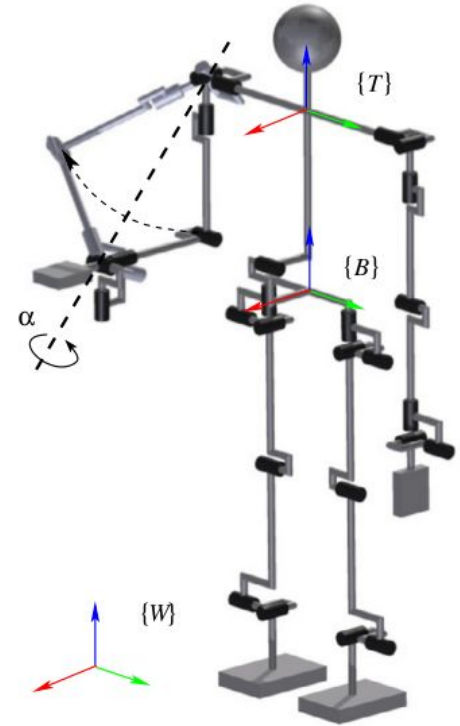
Gyro

$$q_{\text{eyes}_{t+1}} = q_{\text{eyes}_t} + \Delta T \left(G \cdot J^{\#} \cdot \left(FP_d - K_{FP}\left(q_{\text{eyes}_t}\right) \right) - \dot{q}_c \right)$$
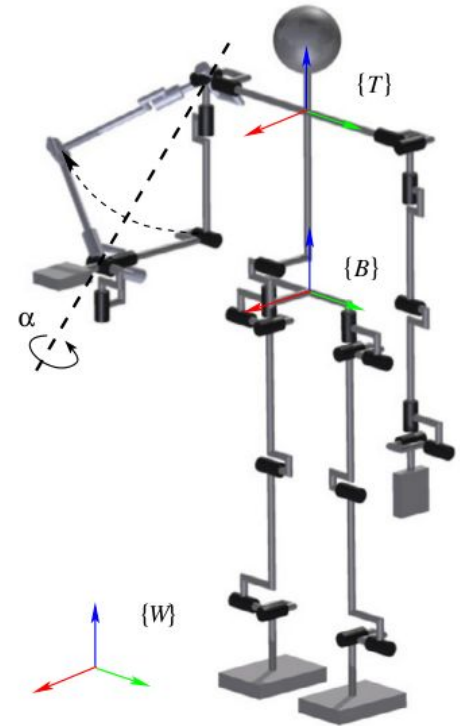
Inverse kinematics solution under multiple task constraints

- For arm reach of a humanoid, whole-body motion motion may be employed.
- Total number of joints, e.g.: $n_{total} = n_{leg} + n_{torso} + n_{arm} = 6 + 1 + 7 = 17$.
- Degree of kinematic redundancy = 17 - 6 = **11**.
- With such a high degree of redundancy, it is possible to realize multiple additional tasks.

Section 2.8 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.
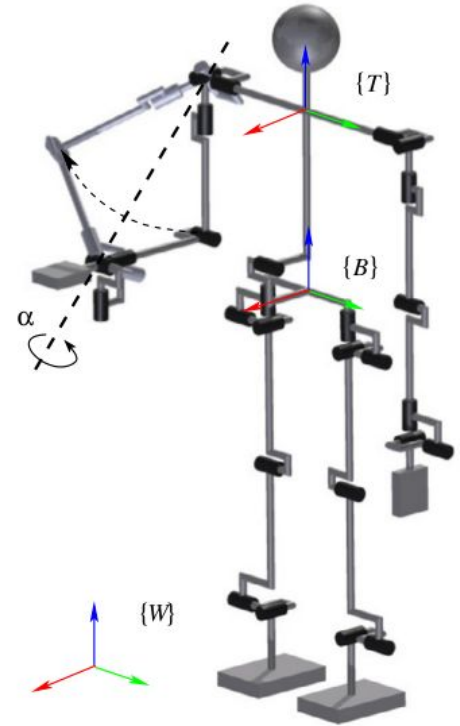
Motion task constraints

- Main task: hand motion ~ reaching.
- Other motion tasks:
  - keep balance
  - avoid obstacles
  - avoid self-collisions
  - avoid singularities
  - avoid joint limits
  - gaze task
  - …
- Constraints helpful in resolving kinematic redundancy.
  - Caveat: Overconstrained state (task conflict).

Section 2.8.1 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.
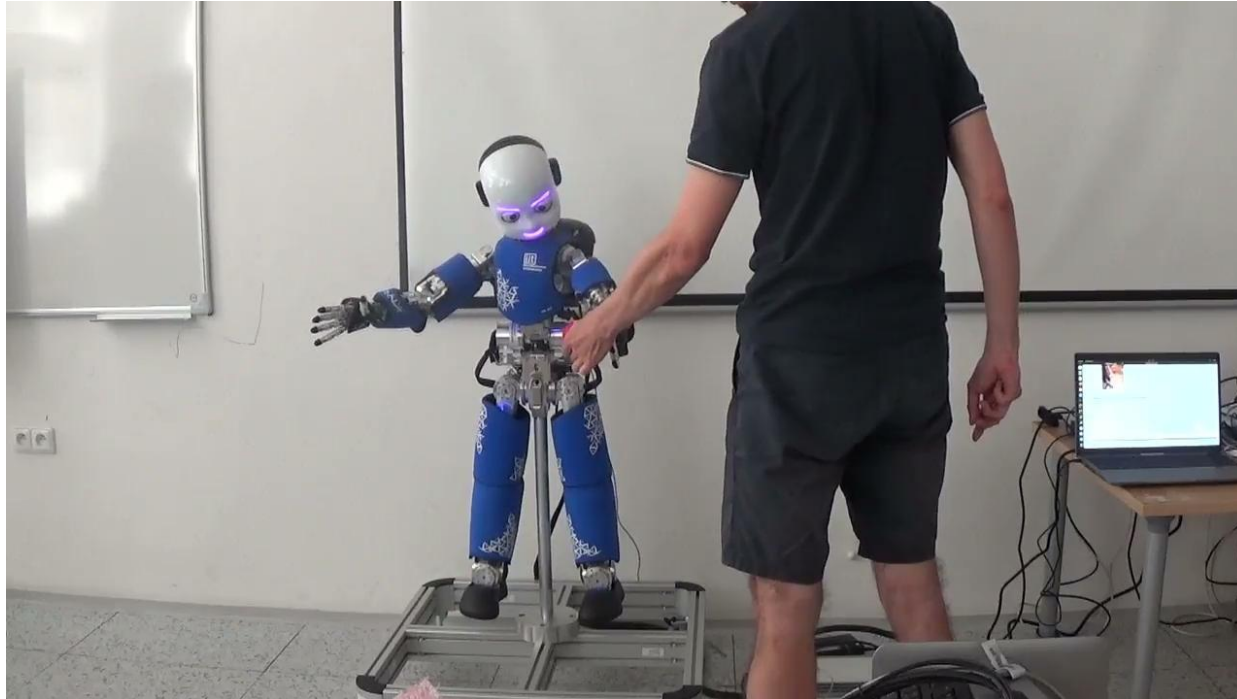
Types of motion task constraints

- link-motion constraints
  - movement of reaching hand
  - movement of Center of Mass
- joint-motion constraints
  - joint ranges
  - joint velocity limits
  - singularity avoidance
- equality- and inequality (unilateral)-type constraints
- permanently active and temporal constraints
- high-priority and low-priority constraints

Section 2.8.1 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.

Examples – Classify

- joint limits
  - inequality, permanent, high-priority
- reaching task
  - inequality / minimization term, middle priority
- obstacles
  - inequality, temporal, high priority
- singularities
  - inequality, temporal, middle priority
- postural constraints
  - min. term / inequality, permanent, low-priority

- equality- and inequality (unilateral)-type constraints
- permanently active and temporal constraints
- high-priority and low-priority constraints

Section 2.8.1 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.

iCub red ball demo – IK solver (+ Cartesian controller) + gaze controller

Pattacini, U., Nori, F., Natale, L., Metta, G., & Sandini, G. (2010, October). An experimental evaluation of a novel minimum-jerk Cartesian controller for humanoid robots. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1668-1674). IEEE.

Roncone, A., Pattacini, U., Metta, G., & Natale, L. (2016, June). A Cartesian 6-DoF Gaze Controller for Humanoid Robots. In *Robotics: Science and Systems* (Vol. 2016).

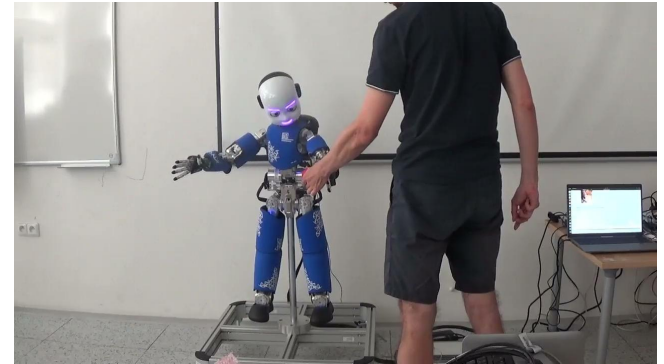# iCub IK solver (+ Cartesian controller) + gaze controller

https://github.com/robotology/icub-basic-demos

- Task 1 - Reach
  - Task DoFs: m = 6
    - https://github.com/robotology/icub-basic-demos/blob/master/demoRedBall/src/main.cpp#L147
  - Joint DoFs: ?
    - m = 7 per arm + 2 of the torso = 9
    - https://github.com/robotology/icub-basic-demos/blob/master/demoRedBall/src/main.cpp#L136
- Task 2 - Gaze
  - Task DoFs: m = 3
  - Joint DoFs: n = 6
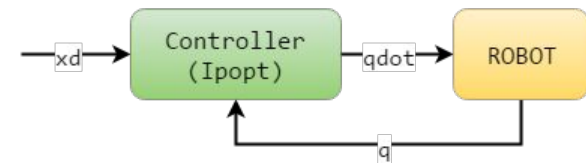 - https://github.com/robotology/icub-basic-demos/blob/master/demoRedBall/src/main.cpp#L898

Pattacini, U., Nori, F., Natale, L., Metta, G., & Sandini, G. (2010, October). An experimental evaluation of a novel minimum-jerk Cartesian controller for humanoid robots. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1668-1674). IEEE.

Roncone, A., Pattacini, U., Metta, G., & Natale, L. (2016, June). A Cartesian 6-DoF Gaze Controller for Humanoid Robots. In *Robotics: Science and Systems* (Vol. 2016).
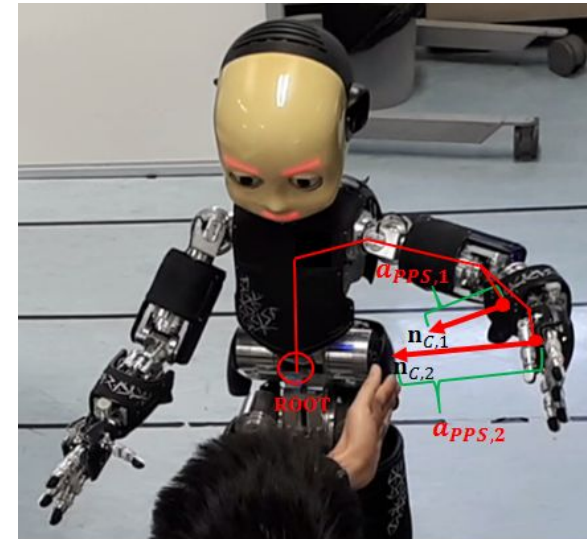
Reactive controller

$$\dot{\mathbf{q}}_d = \arg \min_{\dot{\mathbf{q}} \in \mathbb{R}^n} \left(\left\| \overline{\mathbf{x}}_{EEd} - \left(\overline{\mathbf{x}}_{EE} + T_S \mathbf{J}(\overline{\mathbf{q}}) \dot{\mathbf{q}} \right) \right\|^2 \right)$$

$$\text{s.t.} \begin{cases} \mathbf{q}_L < \overline{\mathbf{q}} + T_S \dot{\mathbf{q}} < \mathbf{q}_U \\ \dot{\mathbf{q}}_L < \dot{\mathbf{q}} < \dot{\mathbf{q}}_U \\ \text{other constraints...} \end{cases}$$

$$\begin{cases} \mathbf{s} = -\mathbf{J}_C^T \mathbf{n}_C \cdot V_C \cdot a_{PPS} \\ \dot{q}_{L,i} = \max\left\{ V_{L,i}, s_i \right\} \quad s_i > 0 \\ \dot{q}_{U,i} = \min\left\{ V_{U,i}, s_i \right\} \quad s_i < 0 \end{cases}$$

obstacles →

- link-motion constraints
  - movement of reaching hand
  - movement of Center of Mass
- joint-motion constraints
  - joint ranges
  - joint velocity limits
  - singularity avoidance
- equality- and inequality (unilateral)-type constraints
- permanently active and temporal constraints
- high-priority and low-priority constraints

81

Conflict resolution for multiple motion task constraints

- Inverse kinematics with constraints in a **hierarchical structure**, using null space projections
- How to set priorities? (Sentis & Khatib 2005)
  - joint motion constraints
  - link-motion constraints
    - balance - CoM control
    - hand position
  - postural-variation constraints
- Pros: Stability may be guaranteed.
- Cons: Difficulty of incorporating inequality constraints.

- Inverse kinematics as multiobjective optimization problem
- Pros: All kinds of constraints, incl. inequality constraints, can be incorporated, even on the run.
- Cons: Difficult to guarantee control stability.

Section 2.8.1 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.
L. Sentis, O. Khatib, Synthesis of whole-body behaviors through hierarchical control of behavioral primitives, International Journal of Humanoid Robotics 02 (2005) 505–518.

iCub Red Ball demo ++

https://youtu.be/NMzhDqVgVvk

Piga, N., Onyshchuk, Y., Pasquale, G., Pattacini, U., and Natale, L., ROFT: Real-time Optical Flow-aided 6D Object Pose and Velocity Tracking, IEEE Robotics & Automation Magazine, vol. 7, no. 1, pp. 159-166, 2022.

iCub balancing and reaching
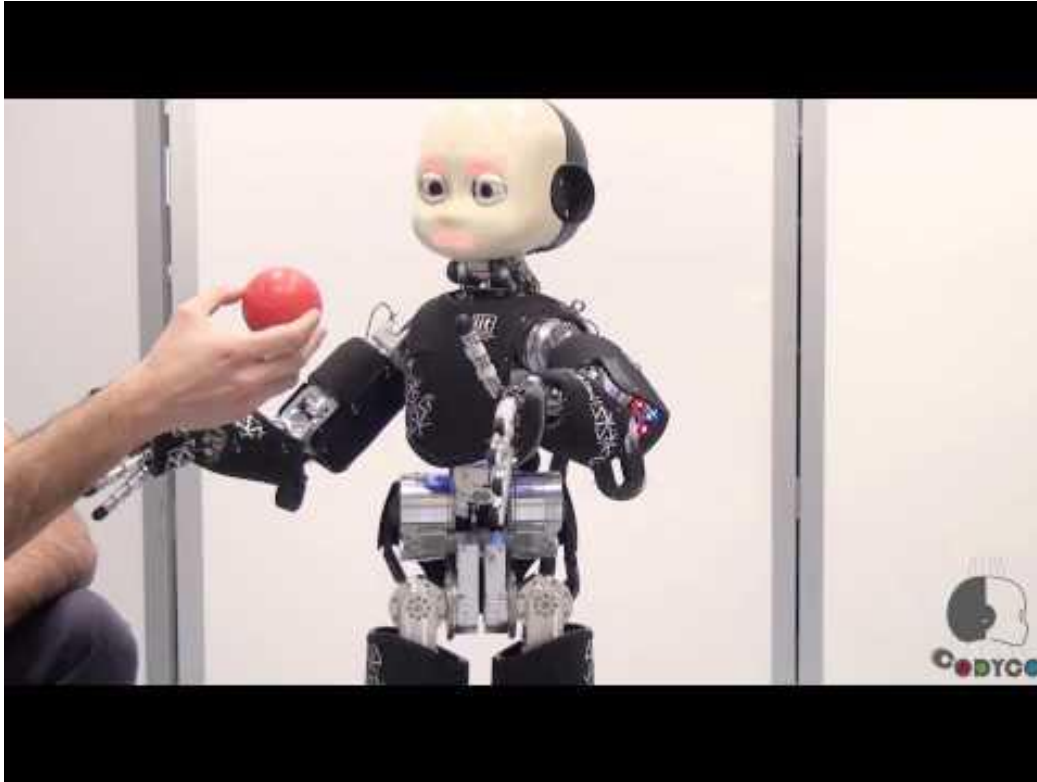
Set of admissible tasks:

- right foot wrench task - regulate the right foot interaction wrench to a predefined value
- left foot wrench
- right arm wrench
- left arm wrench
- postural task
- reaching task
- gaze task

(N.B. wrench is the analog of twist for forces - 6D "force")

https://youtu.be/7CxaynVnsCI

Nori, F., Traversaro, S., Eljaik, J., Romano, F., Del Prete, A., & Pucci, D. (2015). iCub whole-body control through force regulation on rigid non-coplanar contacts. *Frontiers in Robotics and AI*, *2*, 6. https://www.frontiersin.org/articles/10.3389/frobt.2015.00006/full

Wrap–up

- Kinematic redundancy
- Singular configurations
- Manipulability
- Self motion / Null space
- Inverse kinematics
  - Analytical / closed-form
  - Iterative - Jacobian (pseudo)inverse
  - Optimization without Jacobian inverse

Next week

- Demo time at E210. Please come to the lecture E301 as usual (12:45) and the TAs will pick you up here.
- Labs will take place as usual at E132.

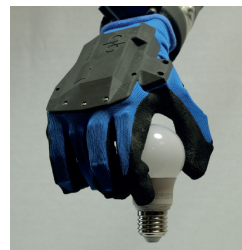# Robots - humanoids and cobots
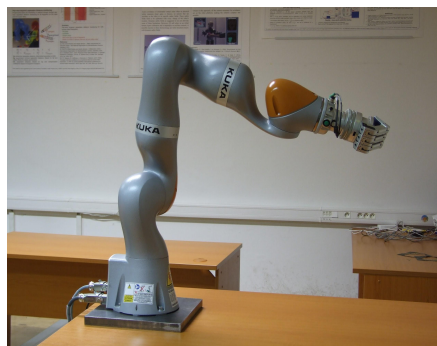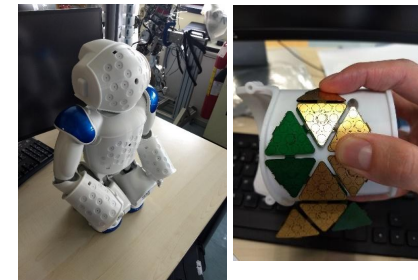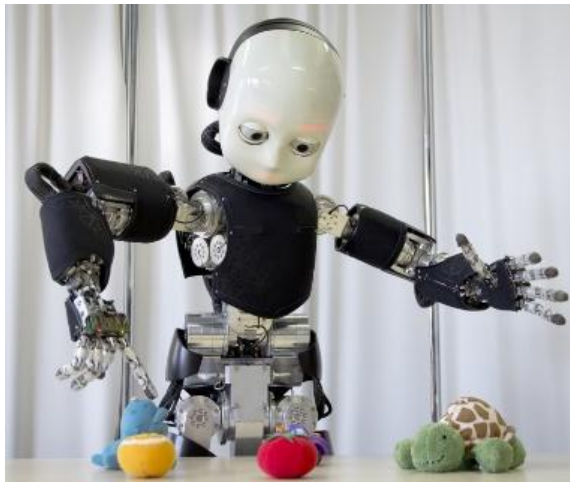
KUKA LBR iiwa

Kinova Gen3

UR10 + Airskin

Robot hands and grippers

Anthropomorphic hands

Barrett Hand
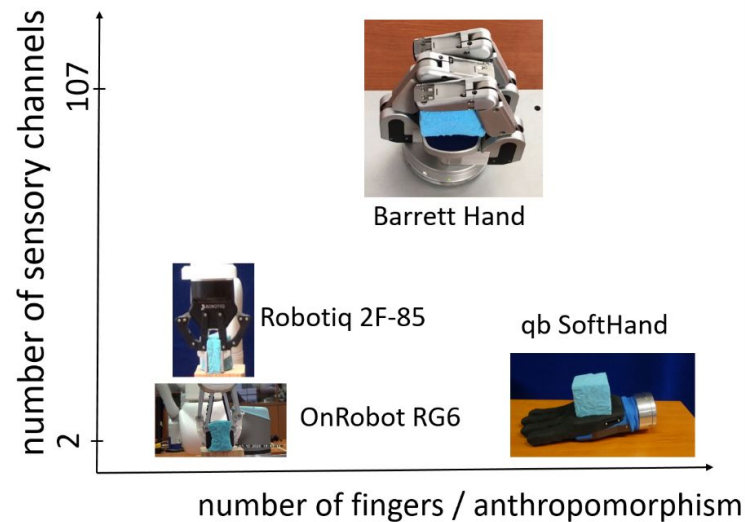(96 tactile + 3 fingertip joint torque + 8 joint pos. sensors)

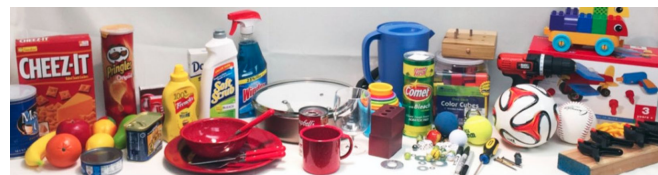Qb SoftHand (1 motor with position and current sensor)

Industrial parallel jaw 2-finger grippers
Robotiq 2F-85, OnRobot RG6

iCub hand

number of sensory channels: 107, 2

Barrett Hand

Robotiq 2F-85

qb SoftHand

OnRobot RG6

number of fingers / anthropomorphism

YCB object and model set

deformable objects set

Resources

- Books
  - Sections 2.4.2 - 2.7.3 in Nenchev, D. N., Konno, A., & Tsujita, T. (2018). Humanoid robots: Modeling and control. Butterworth-Heinemann.
  - Ch.5 Velocity kinematics and statics in Lynch, K. M., & Park, F. C. (2017). Modern robotics. Cambridge University Press. (see also https://youtu.be/6tj8QLF69Ok).
  - Part III in Corke, P. I. (2013). Robotics, vision and control: fundamental algorithms in MATLAB Berlin: Springer.
- Online resources
  - Howie Choset: https://www.slideserve.com/antonia/inverting-the-jacobian-and-manipulability
  - Modern robotics (Lynch and Park)
    - https://modernrobotics.northwestern.edu/nu-gm-book-resource/velocity-kinematics-and-statics/
  - http://handbookofrobotics.org/view-chapter/videodetails/10
- Tutorials
  - Haviland, J., & Corke, P. (2022). Manipulator Differential Kinematics
    - Part I: Kinematics, Velocity, and Applications. https://arxiv.org/pdf/2207.01796.pdf
    - Part II: Acceleration and Advanced Applications. https://arxiv.org/pdf/2207.01794.pdf
- Articles
  - Chiaverini, S., Oriolo, G., & Maciejewski, A. A. (2016). Redundant robots. In *Springer Handbook of Robotics* (pp. 221-242). Springer, Cham.
  - Lloyd, S., Irani, R. A., & Ahmadi, M. (2022). Fast and Robust Inverse Kinematics of Serial Robots Using Halley's Method. IEEE Transactions on Robotics, 38(5), 2768-2780.
  - Pattacini, U., Nori, F., Natale, L., Metta, G., & Sandini, G. (2010, October). An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *2010 IEEE/RSJ international conference on intelligent robots and systems* (pp. 1668-1674). IEEE.