Jacobians & Robot Sensing

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 3

August 9, 2017

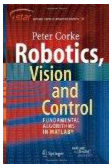
metr4202-staff@itee.uq.edu.au

http://robotics.itee.uq.edu.au/~metr4202/ [http://**metr4202.com**]

Lecture Schedule

Week	Date	Lecture (W: 3:05p-4:50, 7-222)
1	26-Jul	Introduction Representing Position & Orientation & State
2	2-Aug	Robot Forward Kinematics (Frames, Transformation Matrices & Affine Transformations)
3	9-Aug	Robot Inverse Kinematics & Dynamics (Jacobians)
4	16-Aug	*Ekka Day* (Robot Kinematics & Kinetics Review)
5	23-Aug	Jacobians & Robot Sensing Overview
6	30-Aug	Robot Sensing: Single View Geometry & Lines
7	6-Sep	Robot Sensing: Multiple View Geometry
8	13-Sep	Robot Sensing: Feature Detection
9	20-Sep	***Mid-Semester Exam***
	27-Sep	*Study break*
10	4-Oct	Motion Planning
11	11-Oct	Probabilistic Robotics: Localization & SLAM
12	18-Oct	Probabilistic Robotics: Planning & Control (State-Space/Shaping the Dynamic Response/LQR)
13	25-Oct	The Future of Robotics/Automation + Challenges + Course Review

Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook:
364220144X

Today

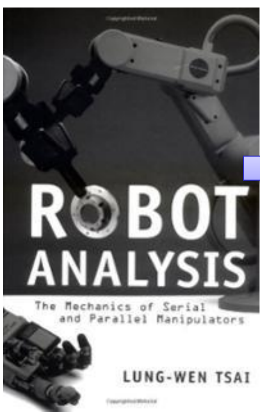
➔ Dynamics & Control ←

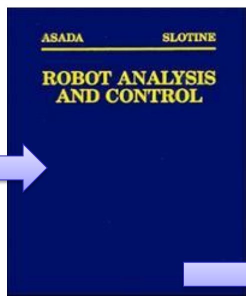
- RVC
  - Chapter 9: Dynamics & Control
  - Note that:
    - "Jacobian + Vision = Visual Servoing" (Chapter 15)
    - See Eqs. **15.6** and **16.15**

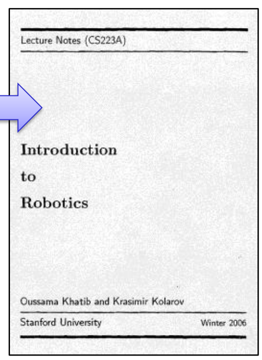
- Multiple View Geometry
  - RVC
    §14.2: Robot Arm Kinematics

Next Time

Reference Material







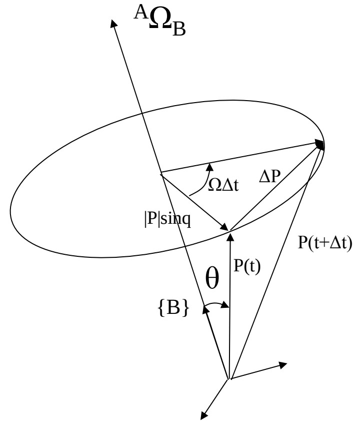

```
On class webpage
Password: metr4202
```

Robot Dynamics

Robot Dynamics

Angular Velocity

- If we look at a small timeslice as a frame rotates with a moving point, we find



$$
\begin{aligned}
|\Delta \mathbf{P}| &= \left(\left|{}^A\mathbf{P}\right| \sin\theta\right)\left(\left|{}^A\Omega_B\right|\Delta t\right)\\
\frac{|\Delta \mathbf{P}|}{\Delta t} &= \left(\left|{}^A\mathbf{P}\right| \sin\theta\right)\left(\left|{}^A\Omega_B\right|\right)\\
&= {}^A\Omega_B \times {}^A\mathbf{P}
\end{aligned}
$$

$$
{}^A\mathbf{V}_P = {}^A\Omega_B \times {}^A R_B {}^B\mathbf{P}
$$

Velocity

- Recall that we can specify a point in one frame relative to another as

$$
{}^A\mathbf{P} = {}^A\mathbf{P}_B + {}^A_B\mathbf{R}\,{}^B\mathbf{P}
$$

- Differentiating w/r/t to **t** we find

$$
\begin{aligned}
{}^A\mathbf{V}_P = \frac{d}{dt}{}^A\mathbf{P} &= \lim_{\Delta t \to 0} \frac{{}^A\mathbf{P}(t+\Delta t) - {}^A\mathbf{P}(t)}{\Delta t}\\
&= {}^A\dot{\mathbf{P}}_B + {}^A_B\mathbf{R}\,{}^B\dot{\mathbf{P}} + {}^A_B\dot{\mathbf{R}}\,{}^B\mathbf{P}
\end{aligned}
$$

- This can be rewritten as

$$
{}^A\mathbf{V}_P = {}^A\mathbf{V}_{BORG} + {}^A\mathbf{R}_B {}^B\mathbf{V}_P + {}^A\Omega_B \times {}^A\mathbf{R}_B {}^B\mathbf{P}
$$

Skew – Symmetric Matrix

$$\mathbf{V} = \omega \times \mathbf{r}$$

$$\Omega = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

$$\rightarrow \mathbf{V} = \Omega \mathbf{r}$$

Velocity Representations

- Euler Angles
  - For Z-Y-X (α,β,γ):

$$\begin{pmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{pmatrix} = \begin{pmatrix} -S\beta & 0 & 1 \\ C\beta S\gamma & C\gamma & 0 \\ C\beta C\gamma & -S\beta & 0 \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

- Quaternions

$$\begin{pmatrix} \dot{\varepsilon}_0 \\ \dot{\varepsilon}_1 \\ \dot{\varepsilon}_2 \\ \dot{\varepsilon}_3 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \varepsilon_1 & -\varepsilon_2 & -\varepsilon_3 \\ \varepsilon_0 & \varepsilon_3 & -\varepsilon_2 \\ -\varepsilon_3 & \varepsilon_0 & \varepsilon_1 \\ \varepsilon_2 & -\varepsilon_1 & \varepsilon_0 \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

Manipulator Velocities

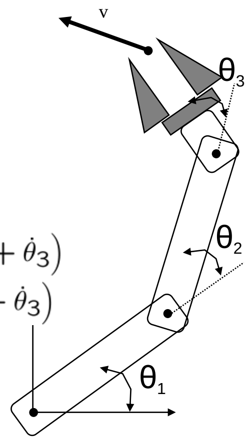
- Consider again the schematic of the planar manipulator shown. We found that the end effector position is given by

$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) + L_3 \cos(\theta_1 + \theta_2 + \theta_3)$$
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) + L_3 \sin(\theta_1 + \theta_2 + \theta_3)$$

- Differentiating w/r/t to t

$$\dot{x} = -L_1\,\mathsf{s}_1\,\dot{\theta}_1 - L_2\,\mathsf{s}_{12}\left(\dot{\theta}_1 + \dot{\theta}_2\right) - L_3\,\mathsf{s}_{123}\left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right)$$
$$\dot{y} = L_1\,\mathsf{c}_1\,\dot{\theta}_1 + L_2\,\mathsf{c}_{12}\left(\dot{\theta}_1 + \dot{\theta}_2\right) + L_3\,\mathsf{c}_{123}\left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right)$$

- This gives the end effector velocity as a function of pose and joint velocities



Manipulator Velocities [2]

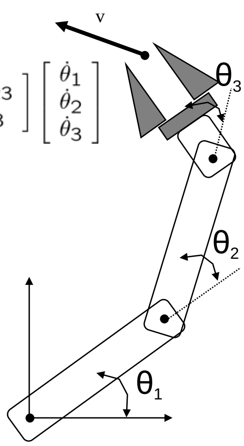
- Rearranging, we can recast this relation in matrix form

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -L_1\,\mathsf{s}_1 - L_2\,\mathsf{s}_{12} - L_3\,\mathsf{s}_{123} & -L_2\,\mathsf{s}_{12} - L_3\,\mathsf{s}_{123} & -L_3\,\mathsf{s}_{123} \\ L_1\,\mathsf{c}_1 + L_2\,\mathsf{c}_{12} + L_3\,\mathsf{c}_{123} & L_2\,\mathsf{c}_{12} + L_3\,\mathsf{c}_{123} & L_3\,\mathsf{c}_{123} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

- Or

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \frac{\partial x}{\partial \theta_3} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \frac{\partial y}{\partial \theta_3} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

- The resulting matrix is called the Jacobian and provides us with a mapping from Joint Space to Cartesian Space.



Moving On…Differential Motion

- Transformations also encode differential relationships
- Consider a manipulator (say 2DOF, RR)

$$x(\theta_1, \theta_2) = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$
$$y(\theta_1, \theta_2) = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$

- Differentiating with respect to the **angles** gives:

$$dx = \frac{\partial x(\theta_1, \theta_2)}{\partial \theta_1} d\theta_1 + \frac{\partial x(\theta_1, \theta_2)}{\partial \theta_2} d\theta_2$$

$$dy = \frac{\partial y(\theta_1, \theta_2)}{\partial \theta_1} d\theta_1 + \frac{\partial y(\theta_1, \theta_2)}{\partial \theta_2} d\theta_2$$

Differential Motion [2]

- Viewing this as a matrix → Jacobian

$$d\mathbf{x} = J d\theta$$

$$J = \begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix}$$

$$J = \begin{bmatrix} [J_1] & [J_2] \end{bmatrix}$$

$$v = J_1 \dot{\theta}_1 + J_2 \dot{\theta}_2$$

Infinitesimal Rotations

- $\cos(d\phi) = 1, \sin(d\phi) = d\phi$

$$\mathbf{R}_x(d\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cd\phi & -sd\phi \\ 0 & sd\phi & cd\phi \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -d\phi_x \\ 0 & d\phi_x & 1 \end{bmatrix}$$

$$\mathbf{R}_y(d\phi) = \begin{bmatrix} cd\phi & 0 & sd\phi \\ 0 & 1 & 0 \\ -sd\phi & 0 & cd\phi \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & d\phi_y \\ 0 & 1 & 0 \\ -d\phi_y & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_z(d\phi) = \begin{bmatrix} cd\phi & -sd\phi & 0 \\ sd\phi & cd\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & -d\phi_z & 0 \\ d\phi_z & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Note that:

$$R_x(d\varphi)R_y(d\varphi) = R_y(d\varphi)R_x(d\varphi)$$

→ Therefore … they **commute**

The Jacobian

- In general, the Jacobian takes the form (for example, **j joints** and in **i operational space**)

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_i \end{bmatrix} = \begin{bmatrix} \frac{\partial x_1}{\partial \theta_1} & \frac{\partial x_1}{\partial \theta_2} & \cdots & \frac{\partial x_1}{\partial \theta_j} \\ \frac{\partial x_2}{\partial \theta_1} & \frac{\partial x_2}{\partial \theta_2} & \cdots & \frac{\partial x_2}{\partial \theta_j} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_i}{\partial \theta_1} & \frac{\partial x_i}{\partial \theta_2} & \cdots & \frac{\partial x_i}{\partial \theta_j} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_j \end{bmatrix}$$

- Or more succinctly

$$\dot{\mathbf{X}} = \mathbf{J}(\theta)\dot{\theta}$$

Jacobian [2]

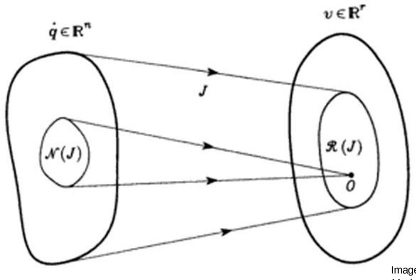


Image:. Sciavicco and Siciliano, *Modelling and Control of Robot Manipulators*, 2nd ed, 2000

- Jacobian can be viewed as a mapping from Joint velocity space ($\dot{q}$ to Operational velocity space (v)

Revisiting The Jacobian

- I told you:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_i \end{bmatrix} = \begin{bmatrix} \frac{\partial x_1}{\partial \theta_1} & \frac{\partial x_1}{\partial \theta_2} & \cdots & \frac{\partial x_1}{\partial \theta_j} \\ \frac{\partial x_2}{\partial \theta_1} & \frac{\partial x_2}{\partial \theta_2} & \cdots & \frac{\partial x_2}{\partial \theta_j} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_i}{\partial \theta_1} & \frac{\partial x_i}{\partial \theta_2} & \cdots & \frac{\partial x_i}{\partial \theta_j} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_j \end{bmatrix}$$

- True, but we can be more “explicit”

Jacobian: **Explicit Form**

- For a serial chain (robot): The velocity of a link with respect to the proceeding link is dependent on the type of link that connects them
- If the joint is **prismatic** (**ϵ=1**), then $\mathbf{v}_i = \frac{dz}{dt}$
- If the joint is **revolute** (**ϵ=0**), then $\omega = \frac{d\theta}{dt}$ (in the $\hat{k}$ direction)

$$\therefore v = \sum_{i=1}^{N}\left(\varepsilon_i v_i + \bar{\varepsilon}_i\left(\omega_i \times \mathbf{p}_{i-1}^{i}\right)\right) \qquad \omega = \sum_{i=1}^{N}\left(\bar{\varepsilon}_i\left(\dot{\boldsymbol{\theta}}_i\right)\right) = \sum_{i=1}^{N}\left(\bar{\varepsilon}_i \mathbf{z}_i\left(\dot{\theta}_i\right)\right)$$

$$\rightarrow v = J_v \dot{\mathbf{q}} \qquad \boldsymbol{\omega} = J_\omega \dot{\mathbf{q}}$$

- Combining them (with **v**=(Δx, Δθ))

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

Jacobian: **Explicit Form [2]**

- The overall Jacobian takes the form

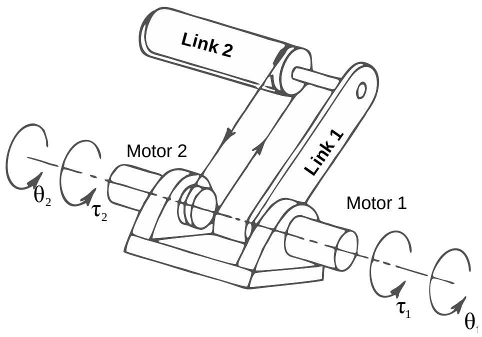
$$J = \begin{bmatrix} \frac{\partial x_P}{\partial q_1} & \cdots & \frac{\partial x_P}{\partial q_n} \\ \bar{\varepsilon}_1 z_1 & \cdots & \bar{\varepsilon}_1 z_n \end{bmatrix}$$

- The Jacobian for a particular frame (F) can be expressed:

$${}^{F}J = \begin{bmatrix} {}^{F}J_v \\ {}^{F}J_\omega \end{bmatrix} = \begin{bmatrix} \frac{\partial\, {}^{F}x_P}{\partial q_1} & \cdots & \frac{\partial\, {}^{F}x_P}{\partial q_n} \\ \bar{\varepsilon}_1\, {}^{F}z_1 & \cdots & \bar{\varepsilon}_1\, {}^{F}z_n \end{bmatrix}$$

Where: ${}^{F}\mathbf{z}_i = {}^{F}_{i}R\, {}^{i}\mathbf{z}_i$ & ${}^{i}\mathbf{z}_i = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$

Motivating Example:
Remotely Driven 2DOF Manipulator



- Introduce $Q_1=\tau_1+\tau_2$ and $Q_2=\tau_2$
- Derivation posted to website

Graphic based on Asada & Slotine p. 112

Dynamics

- We can also consider the forces that are required to achieve a particular motion of a manipulator or other body
- Understanding the way in which motion arises from torques applied by the actuators or from external forces allows us to control these motions
- There are a number of methods for formulating these equations, including
  - Newton-Euler Dynamics
  - Langrangian Mechanics

Dynamics of Serial Manipulators

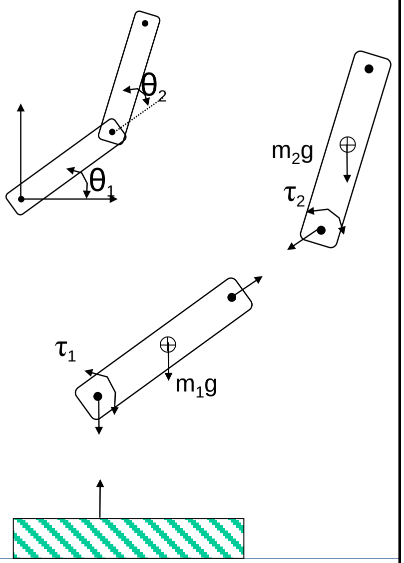
- Systems that keep on manipulating (the system)
- Direct Dynamics:
  - Find the response of a robot arm with torques/forces applied
- Inverse Dynamics:
  - Find the (actuator) torques/forces required to generate a desired trajectory of the manipulator

Dynamics – Newton-Euler

- In general, we could analyse the dynamics of robotic systems using classical Newtonian mechanics

$$\sum F = m\ddot{x}$$
$$\sum T = J\ddot{\theta}$$

- This can entail iteratively calculating velocities and accelerations for each link and then computing force and moment balances in the system
- Alternatively, closed form solutions may exist for simple configurations



Dynamics

- For Manipulators, the general form is

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta)$$

where

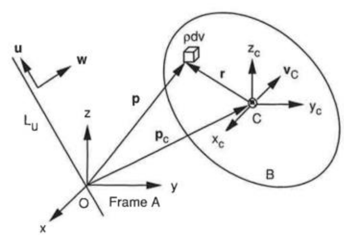
- τ is a vector of joint torques
- Θ is the nx1 vector of joint angles
- M(Θ) is the nxn mass matrix
- $V(\Theta, \dot{\Theta})$is the nx1 vector of centrifugal and Coriolis terms
- G(Θ) is an nx1 vector of gravity terms

- Notice that all of these terms depend on Θ so the dynamics varies as the manipulator move

Dynamics: Inertia

- The moment of inertia (second moment) of a rigid body B relative to a line L that passes through a reference point O and is parallel to a unit vector **u** is given by:

$$I_u^O = \int_V p \times (u \times p)\, \rho dV$$
$$= \int_V \left[p^2 u - \left(p^T u\right) p\right] \rho dV$$



- The scalar product of I^o_u with a second axis (**w**) is called the product of inertia

$$I_{uw}^O = I_u^O \cdot w = \int_V \left[\left(u^T w\right) p^2 - \left(p^T u\right)\left(p^T w\right)\right] \rho dV$$

- If u=w, then we get the moment of inertia:

$$I_{uu} = \int_V \left[p^2 - \left(p^T u\right)^2\right] \rho dV = m r_g^2$$

Where: $\mathbf{r_g}$: *radius of gyration* of B w/r/t to L

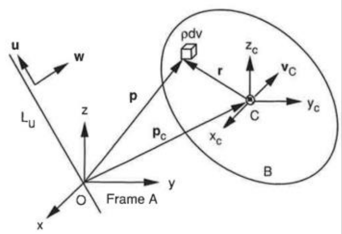
$$r_g = p^2 - \left(p^T u\right)^2 = (u \times p)^2$$

Dynamics: Mass Matrix & Inertia Matrix

- This can be written in a Matrix form as:

$$\mathbf{I}_u^O = I_B^O \mathbf{u}$$

- Where $I^O{}_B$ is the inertial matrix or inertial tensor of the body B about a reference point O



$$I_B^O = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yz} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

- Where to get I_{xx}, etc? ➔ Parallel Axis Theorem

  If CM is the center of mass, then:

$$I_{xx}^O = I_{xx}^{CM} + m\left(y_c^2 + z_c^2\right) \qquad I_{xy}^O = I_{xx}^{CM} + m x_c y_c$$
$$I_{yy}^O = I_{yy}^{CM} + m\left(x_c^2 + z_c^2\right) \qquad I_{yz}^O = I_{xx}^{CM} + m y_c z_c$$
$$I_{zz}^O = I_{zz}^{CM} + m\left(x_c^2 + y_c^2\right) \qquad I_{zx}^O = I_{xx}^{CM} + m z_c x_c$$

Dynamics: Mass Matrix

- The Mass Matrix: Determining via the Jacobian!

$$K = \sum_{i=1}^{N} K_i$$

$$K_i = \frac{1}{2}\left(m_i v_{C_i}^T v_{C_i} + \omega_i^T I_{C_i} \omega_i\right)$$

$$v_{C_i} = J_{v_i}\dot{\theta} \qquad J_{v_i} = \left[\begin{array}{cccccc} \frac{\partial \mathbf{p}_{C_1}}{\partial \theta_1} & \cdots & \frac{\partial \mathbf{p}_{C_i}}{\partial \theta_i} & \underbrace{0}_{i+1} & \cdots & \underbrace{0}_{n} \end{array}\right]$$

$$\omega_i = J_{\omega_i}\dot{\theta} \qquad J_{\omega_i} = \left[\begin{array}{cccccc} \bar{\varepsilon}_1 Z_1 & \cdots & \bar{\varepsilon}_i Z_i & \underbrace{0}_{i+1} & \cdots & \underbrace{0}_{n} \end{array}\right]$$

$$\therefore M = \sum_{i=1}^{N}\left(m_i J_{v_i}^T J_{v_i} + J_{\omega_i}^T I_{C_i} J_{\omega_i}\right)$$

! M is symmetric, positive definite $\therefore m_{ij} = m_{ji}, \dot{\boldsymbol{\theta}}^T M \dot{\boldsymbol{\theta}} > 0$

Dynamics – Langrangian Mechanics

- Alternatively, we can use Langrangian Mechanics to compute the dynamics of a manipulator (or other robotic system)
- The Langrangian is defined as the difference between the Kinetic and Potential energy in the system
- Using this formulation and the concept of virtual work we can find the forces and torques acting on the system.
- This may seem more involved but is often easier to formulate for complex systems

$$L = K - P$$

$$\mathbf{F} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{x}}}\right) - \frac{\partial L}{\partial \mathbf{x}}$$

$$\tau = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta}$$

Dynamics – Langrangian Mechanics [2]

$L = K - P, \dot{\boldsymbol{\theta}}$: Generalized Velocities, M : Mass Matrix

$$\boldsymbol{\tau} = \sum_{i=1}^{N} \tau_i = \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\boldsymbol{\theta}}}\right) - \frac{\partial K}{\partial \boldsymbol{\theta}} + \frac{\partial P}{\partial \boldsymbol{\theta}}$$

$$K = \tfrac{1}{2}\dot{\theta}^T M(\theta)\,\dot{\theta}$$

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\theta}}\right) = \frac{d}{dt}\left(\frac{\partial}{\partial \dot{\theta}}\left(\tfrac{1}{2}\dot{\theta}^T M(\theta)\,\dot{\theta}\right)\right) = \frac{d}{dt}\left(M\dot{\theta}\right) = M\ddot{\theta} + \dot{M}\dot{\theta}$$

$$\rightarrow \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\theta}}\right) - \frac{\partial K}{\partial \theta} = \left[M\ddot{\theta} + \dot{M}\dot{\theta}\right] - \left[\tfrac{1}{2}\dot{\theta}^T M(\theta)\,\dot{\theta}\right] = M\ddot{\theta} + \underbrace{\left\{\dot{M}\dot{\theta} - \tfrac{1}{2}\begin{bmatrix} \dot{\theta}^T \frac{\partial M}{\partial \theta_1}\dot{\theta} \\ \vdots \\ \dot{\theta}^T \frac{\partial M}{\partial \theta_n}\dot{\theta} \end{bmatrix}\right\}}_{\mathbf{v}(\theta,\dot{\theta})}$$

$$\mathbf{v}\left(\theta, \dot{\theta}\right) = \underbrace{C(\theta)\left[\dot{\theta}^2\right]}_{\text{Centrifugal}} + \underbrace{B(\theta)\left[\dot{\theta}\dot{\theta}\right]}_{\text{Coriolis}}$$

$$\Rightarrow \boldsymbol{\tau} = M(\theta)\ddot{\boldsymbol{\theta}} + \mathbf{v}\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right) + \mathbf{g}(\boldsymbol{\theta})$$

Dynamics – Langrangian Mechanics [3]

- The Mass Matrix: Determining via the Jacobian!

$$K = \sum_{i=1}^{N} K_i$$

$$K_i = \frac{1}{2}\left(m_i v_{C_i}^T v_{C_i} + \omega_i^T I_{C_i} \omega_i\right)$$

$$v_{C_i} = J_{v_i}\dot{\theta} \qquad J_{v_i} = \begin{bmatrix} \frac{\partial \mathbf{p}_{C_1}}{\partial \theta_1} & \cdots & \frac{\partial \mathbf{p}_{C_i}}{\partial \theta_i} & \underbrace{0}_{i+1} & \cdots & \underbrace{0}_{n} \end{bmatrix}$$

$$\omega_i = J_{\omega_i}\dot{\theta} \qquad J_{\omega_i} = \begin{bmatrix} \bar{\varepsilon}_1 Z_1 & \cdots & \bar{\varepsilon}_i Z_i & \underbrace{0}_{i+1} & \cdots & \underbrace{0}_{n} \end{bmatrix}$$

$$\therefore M = \sum_{i=1}^{N} \left(m_i J_{v_i}^T J_{v_i} + J_{\omega_i}^T I_{C_i} J_{\omega_i}\right)$$

! M is symmetric, positive definite $\therefore m_{ij} = m_{ji}, \dot{\boldsymbol{\theta}}^T M \dot{\boldsymbol{\theta}} > 0$

Generalized Coordinates

- A significant feature of the Lagrangian Formulation is that any convenient coordinates can be used to derive the system.
- Go from Joint → Generalized
  - Define **p**: $d\mathbf{p} = \mathbf{J}d\mathbf{q}$

$$\mathbf{q} = \begin{bmatrix} q_1 & \dots & q_n \end{bmatrix} \to \mathbf{p} = \begin{bmatrix} p_1 & \dots & p_n \end{bmatrix}$$

➔Thus: the kinetic energy and gravity terms become

$$KE = \tfrac{1}{2}\dot{\mathbf{p}}^T \mathbf{H}^* \dot{\mathbf{p}} \qquad \mathbf{G}^* = \left(\mathbf{J}^{-1}\right)^T \mathbf{G}$$

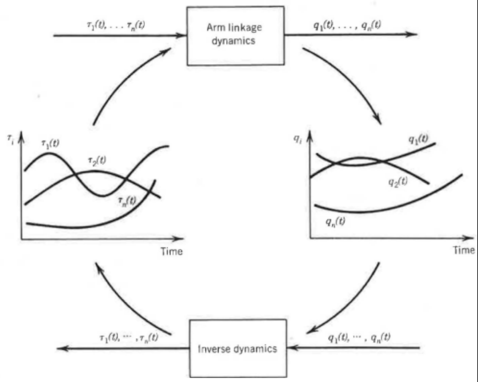
where: $\mathbf{H}^* = \left(\mathbf{J}^{-1}\right)^T \mathbf{H}\mathbf{J}^{-1}$

Inverse Dynamics

- Forward dynamics governs the dynamic responses of a manipulator arm to the input torques generated by the actuators.
- The inverse problem:
  - Going from joint angles to torques
  - Inputs are desired trajectories described as functions of time

    $\mathbf{q} = [q_1 \ \ \dots \ \ q_n] \rightarrow [\theta_1(t) \ \ \theta_2(t) \ \ \theta_3(t)]$
  - Outputs are joint torques to be applied at each instance

    $\boldsymbol{\tau} = [\tau_1 \ \ \dots \ \ \tau_n]$



- Computation "big" (6DOF arm: 66,271 multiplications), but not scary (4.5 ms on PDP11/45)

Graphic from Asada & Slotine p. 119

Also: Inverse Jacobian

- In many instances, we are also interested in computing the set of joint velocities that will yield a particular velocity at the end effector

$$\dot{\theta} = \mathbf{J}(\theta)^{-1}\dot{\mathbf{X}}$$

- We must be aware, however, that the inverse of the Jacobian may be undefined or singular. The points in the workspace at which the Jacobian is undefined are the *singularities* of the mechanism.
- Singularities typically occur at the workspace boundaries or at interior points where degrees of freedom are lost

Inverse Jacobian Example

- For a simple two link RR manipulator:

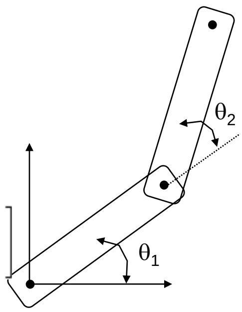
$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

- The Jacobian for this is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -L_1 \mathsf{s}_1 - L_2 \mathsf{s}_{12} & -L_2 \mathsf{s}_{12} \\ L_1 \mathsf{c}_1 + L_2 \mathsf{c}_{12} & L_2 \mathsf{c}_{12} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

- Taking the inverse of the Jacobian yields

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \frac{1}{L_1 L_2 s_2} \begin{bmatrix} L_2 \mathsf{c}_{12} & L_2 \mathsf{s}_{12} \\ -L_1 \mathsf{c}_1 - L_2 \mathsf{c}_{12} & -L_1 \mathsf{s}_1 - L_2 \mathsf{s}_{12} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$



- Clearly, as θ_2 approaches 0 or π this manipulator becomes singular

Static Forces

- We can also use the Jacobian to compute the joint torques required to maintain a particular force at the end effector
- Consider the concept of virtual work

$$F \cdot \delta\mathbf{X} = \tau \cdot \delta\theta$$

- Or

$$F^T \delta\mathbf{X} = \tau^T \delta\theta$$

- Earlier we saw that

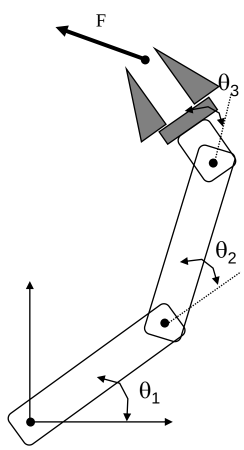
$$\delta\mathbf{X} = \mathbf{J}\delta\theta$$

- So that

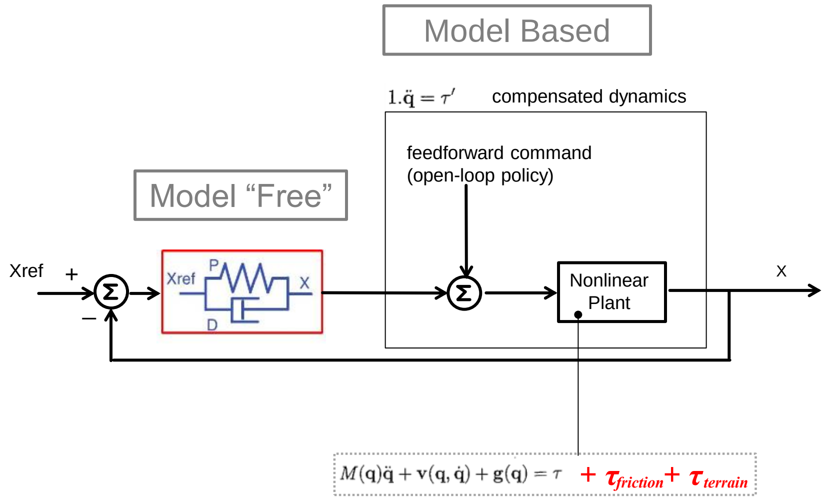
$$F^T \mathbf{J} = \tau^T$$

- Or

$$\tau = \mathbf{J}^T F$$



Operation Space (Computed Torque)

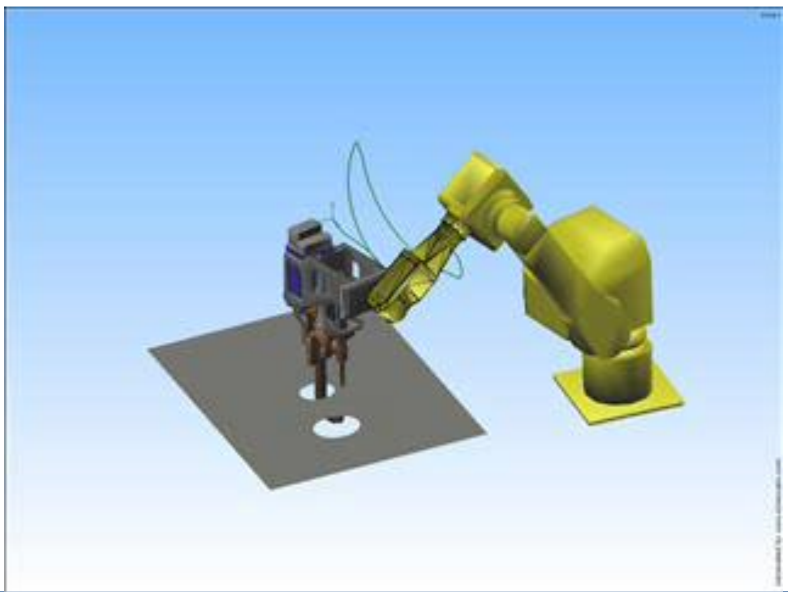


Compensated Manipulation

Dynamics of Parallel Manipulators

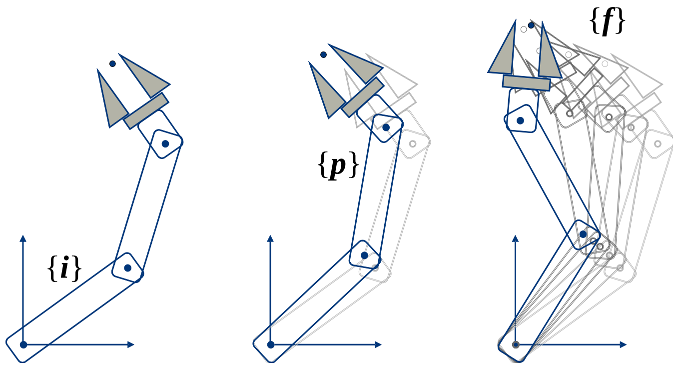
- Traditional Newton-Euler formulation:
  - Equations of motion to be written once for each body of a manipulator
  - Large number of equations
- Lagrangian formulation
  - eliminates all of the unwanted reaction forces and moments at the outset.
  - It is more efficient than the Newton- Euler formulation
  - Numerous constraints imposed by closed loops of a parallel manipulator
- To simplify the problem
  - Lagrangian Multipliers are often introduced
  - Principle of virtual work

Trajectory Generation & Planning



Trajectory Generation

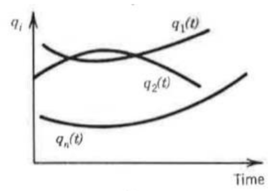
- The goal is to get from an initial position $\{\boldsymbol{i}\}$ to a final position $\{\boldsymbol{f}\}$ via a path points $\{\boldsymbol{p}\}$



Joint Space

Consider only the **joint positions** as a function of time

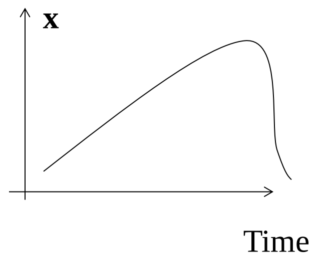
- + Since we control the joints, this is more direct
- -- If we want to follow a particular trajectory, <u>not easy</u>
  - at best lots of intermediate points
  - No guarantee that you can solve the Inverse Kinematics for all path points



Cartesian Workspace

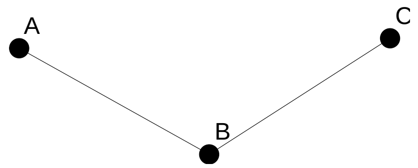
Consider the **Cartesian positions** as a function of time

- \+ Can track shapes exactly
- -- We need to solve the inverse kinematics and dynamics



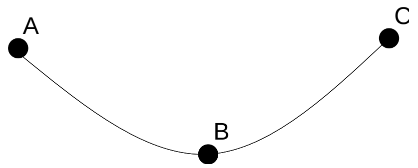
Polynomial Trajectories

- Straight line Trajectories



- Simpler

- Polynomial Trajectories



$$u(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

- Parabolic blends are smoother
- Use "pseudo via points"

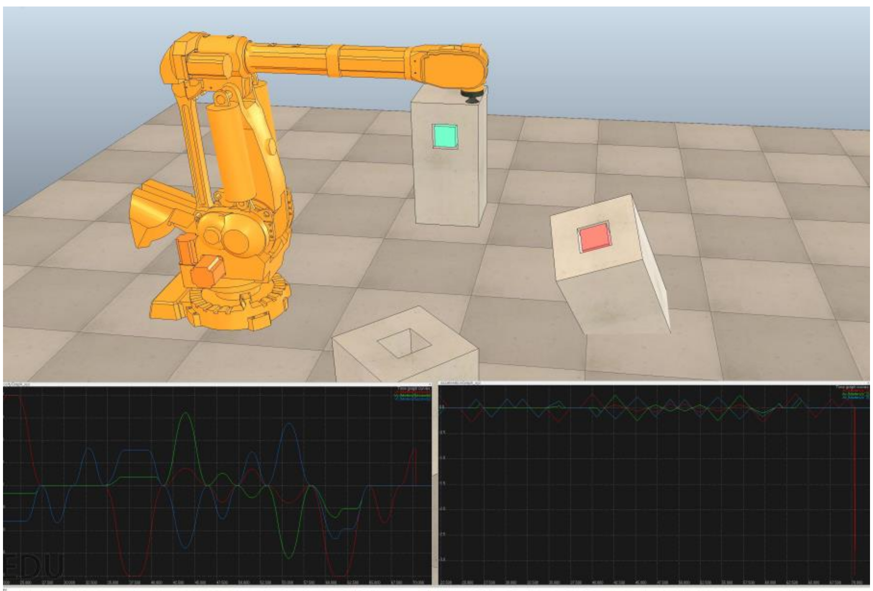
Dynamic Simulation Software

- v-rep
  virtual robot experimentation platform
- Reflexxes

http://www.coppeliarobotics.com/

http://www.reflexxes.com/

CRS: Trajectory Generation & Planning (VREP Example)



Summary

- Kinematics is the study of motion without regard to the forces that create it
- Kinematics is important in many instances in Robotics
- The study of dynamics allows us to understand the forces and torques which act on a system and result in motion
- Understanding these motions, and the required forces, is essential for designing these systems

Sensing: Image Formation / Single-View Geometry

Eclipses, perhaps mythical, also show that 2D Vision is "up to scale" ☺

Source: https://www.flickr.com/groups/nasa-eclipse2017/

Quick Outline

- Frames
- Kinematics

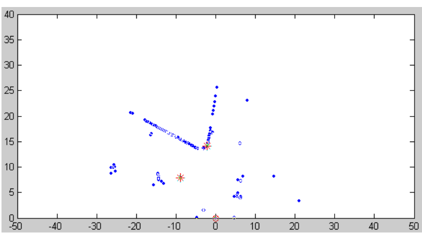
➔ "Sensing Frames" (in space) ➔ Geometry in Vision

1. <u>Perception ➔ Camera Sensors</u>

1. Image Formation
   ➔ "Computational Photography"
2. Calibration
3. Features

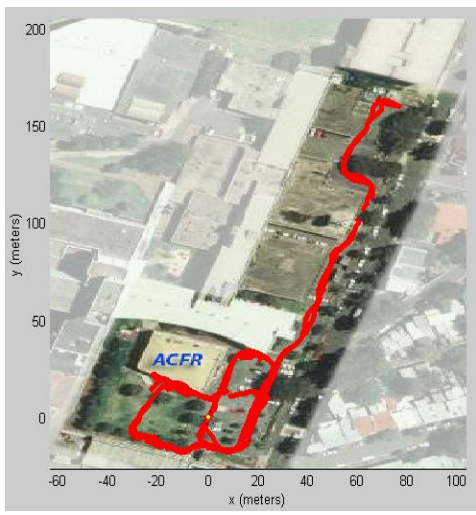
4. Stereopsis and depth
5. Optical flow

Sensor Information: **(not only)** Cameras!



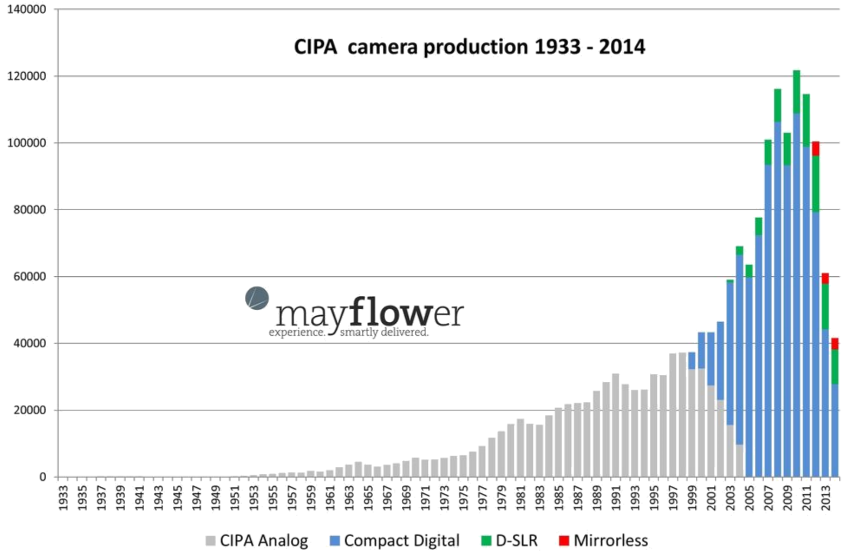
Laser

Vision/Cameras



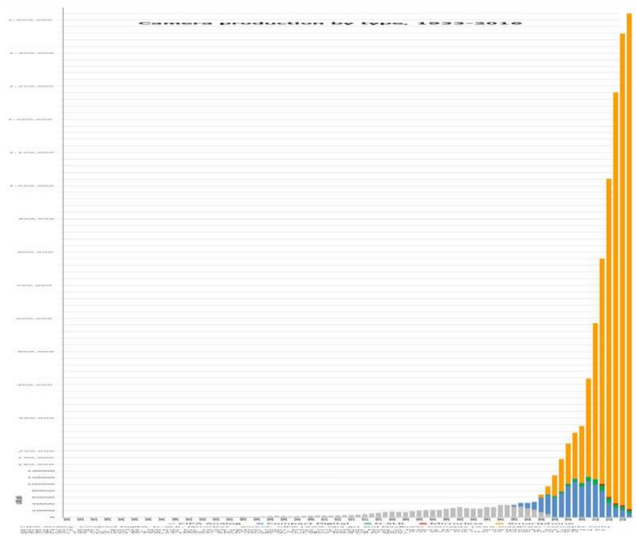
GPS

Okay, Cameras Are Common! [1]



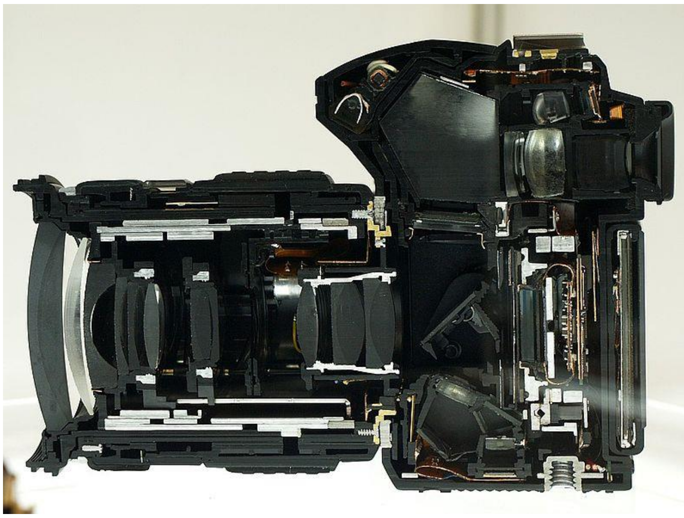
Source: **https://www.mayflower-concepts.com via**
http://www.mirrorlessrumors.com/chart-shows-how-digital-cameras-killed-analog-cameras-and-how-the-selfie-culture-almost-killed-digital-cameras-after-that/

Okay, (Smartphone) Cameras Are Common! [2]



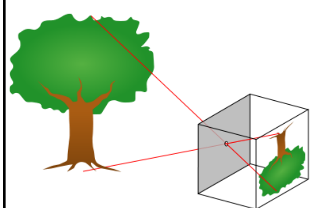
Sources: https://petapixel.com/2017/03/03/latest-camera-sales-chart-reveals-death-compact-camera/
http://www.cipa.jp/stats/dc_e.html

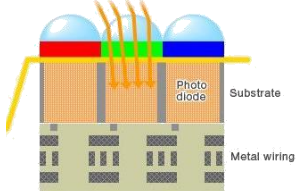
Cameras



Wikipedia, E-30-Cutmodel

Cameras: A 3D ⇒ 2D Photon Counting Sensor*





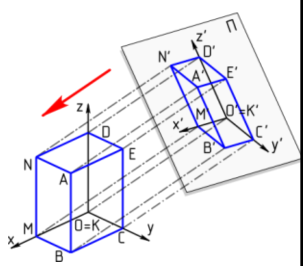
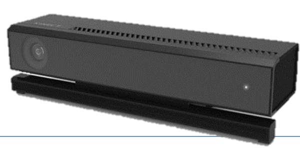
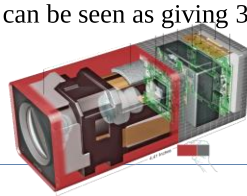


Image Formation ⇒ Image Sensing ⇒ (Re)Projection

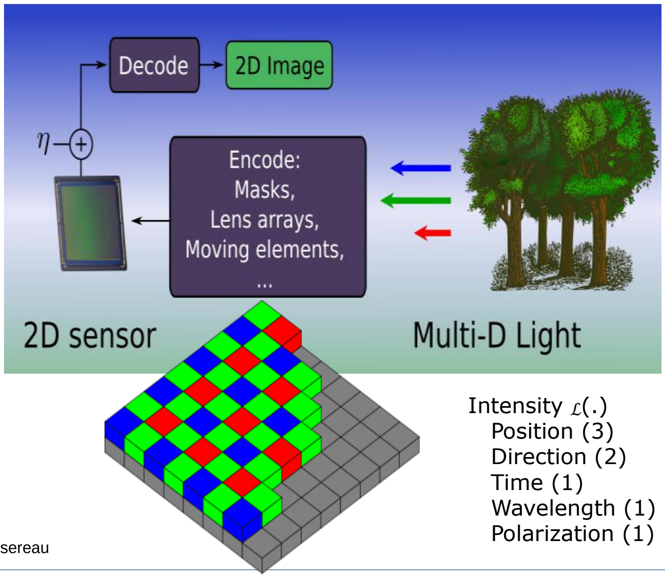
Sources: [Wikipedia, Pinhole Camera](), [Sony sensor](), [Wikipedia, Graphical projection](),

* Well Almost… RGB-D and Light-Field cameras can be seen as giving 3D ⇒ 3D





More Deeply: Part of a **Computational Imaging** Pipeline



Source: Donald Dansereau

Camera Image Formation: 3D↦2D ⇒ **Perspective!**

Camera Image Formation: 3D↦2D ⇒ **Loose Scale!**

Source: https://www.flickr.com/groups/nasa-eclipse2017/

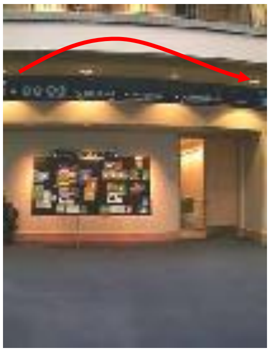
Camera Image Formation “Aberrations”[I]: Lens Optics (Aperture / Depth of Field)

$$N = \frac{f}{\#} = \frac{f}{d}$$

http://en.wikipedia.org/wiki/File:Aperture_in_Canon_50mm_f1.8_II_lens.jpg

Camera Image Formation “Aberrations”[II]: Lens Distortions

Barrel



Pincushion

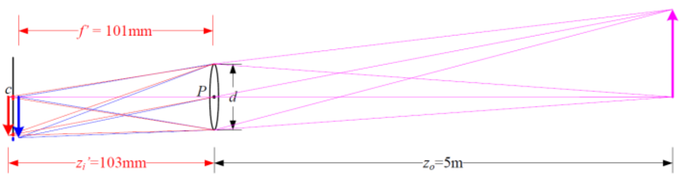
Fisheye

➔ Explore these with `visualize_distortions` in the [Camera Calibration Toolbox]()

Fig. 2.1.3 from Szeliski, *Computer Vision: Algorithms and Applications*

Camera Image Formation “Aberrations” [II]: Lens Optics: Chromatic Aberration

- Chromatic Aberration:

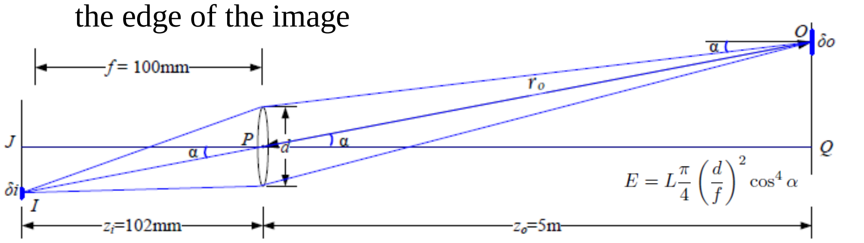


  – In a lens subject to chromatic aberration, light at different wavelengths (e.g., the red and blur arrows) is focused with a different focal length f' and hence a different depth z_i, resulting in both a geometric (in-plane) displacement and a loss of focus

Sec. 2.2 from Szeliski, *Computer Vision: Algorithms and Applications*

Camera Image Formation "Aberrations" [III]: Lens Optics: Vignetting

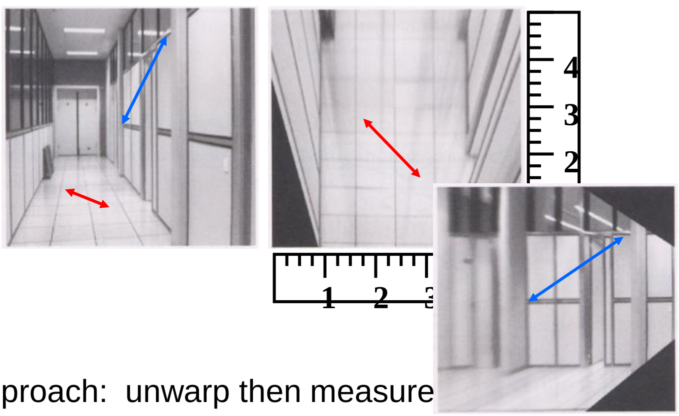
- Vignetting:
  - The tendency for the brightness of the image to fall off towards the edge of the image



  - The amount of light hitting a pixel of surface area δi depends on the square of the ratio of the aperture diameter d to the focal length f, as well as the fourth power of the off-axis angle α, $\cos^4 \alpha$

Sec. 2.2 from Szeliski, *Computer Vision: Algorithms and Applications*

Measurements on Planes
(You can not just add a tape measure!)

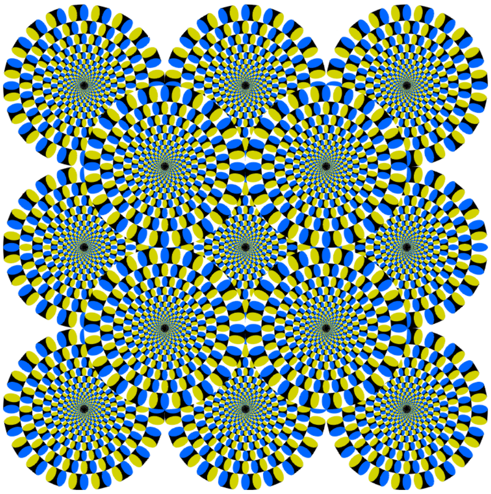


Approach: unwarp then measure

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Perception

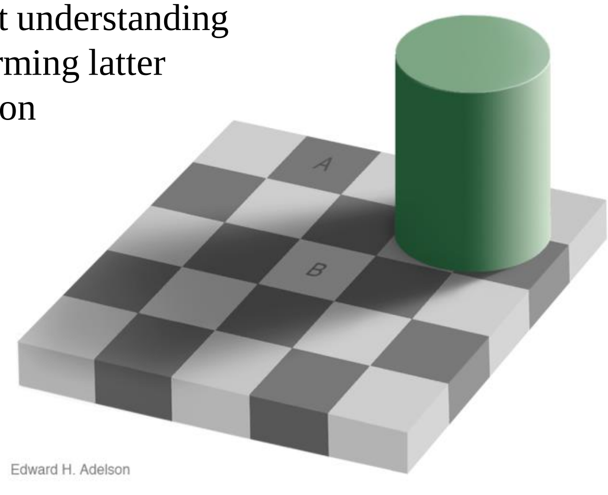
- Making Sense from Sensors



http://www.michaelbach.de/ot/mot_rotsnake/index.html

Perception

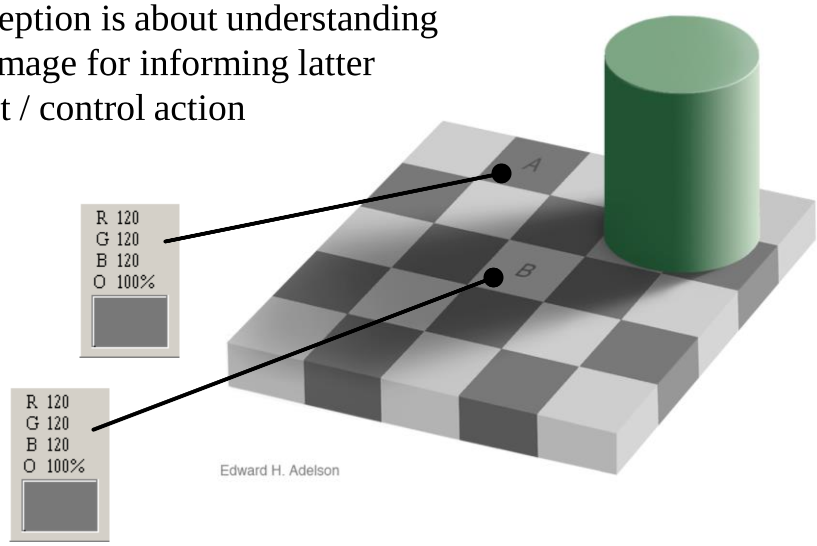
- Perception is about understanding the image for informing latter robot / control action



http://web.mit.edu/persci/people/adelson/checkershadow_illusion.html

Perception

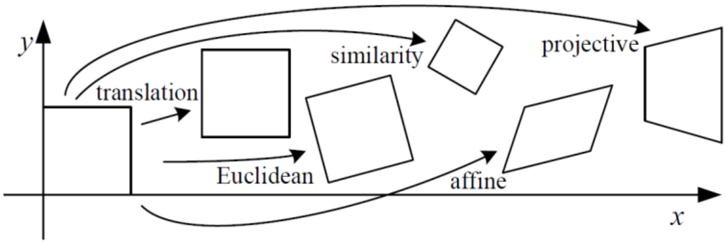
- Perception is about understanding the image for informing latter robot / control action



http://web.mit.edu/persci/people/adelson/checkershadow_illusion.html

Camera MODELS!

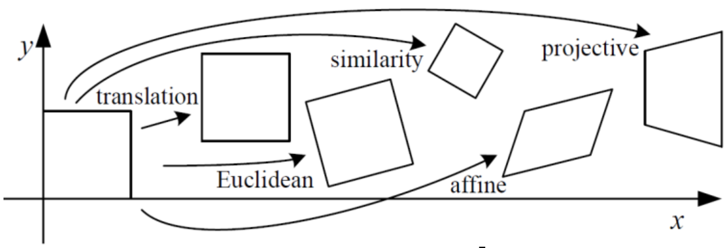
Transformations



- $\boldsymbol{x}'$: New Image & $\underline{\boldsymbol{x}}$: Old Image
- Euclidean: (Distances preserved) $$x' = \begin{bmatrix} R & t \end{bmatrix} \underline{x}$$
- Similarity (Scaled Rotation): (Angles preserved) $$x' = \begin{bmatrix} sR & t \end{bmatrix} \underline{x}$$

Fig. 2.4 from Szeliski, *Computer Vision: Algorithms and Applications*

Transformations [2]



- Affine : (|| lines remain ||) $$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \underline{x}$$
- Projective: (straight lines preserved) H: Homogenous 3x3 Matrix $$x' = \mathbf{H}\underline{x}$$ $$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}$$ $$y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}$$

Fig. 2.4 from Szeliski, Computer Vision: Algorithms and Applications

2-D Transformations

➔ x' = point in the **new** (or 2nd) image

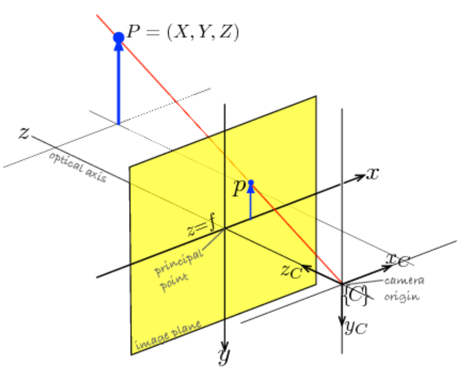
➔ x = point in the old image

- Translation x' = x + t
- Rotation x' = R x + t
- Similarity x' = sR x + t
- Affine x' = A x
- Projective x' = A x

here, x is an inhomogeneous pt (2-vector)

x' is a homogeneous point

Image Formation – Single View Geometry



$$\frac{Y}{Z}=\frac{y}{f} \quad \frac{X}{Z}=\frac{x}{f}$$

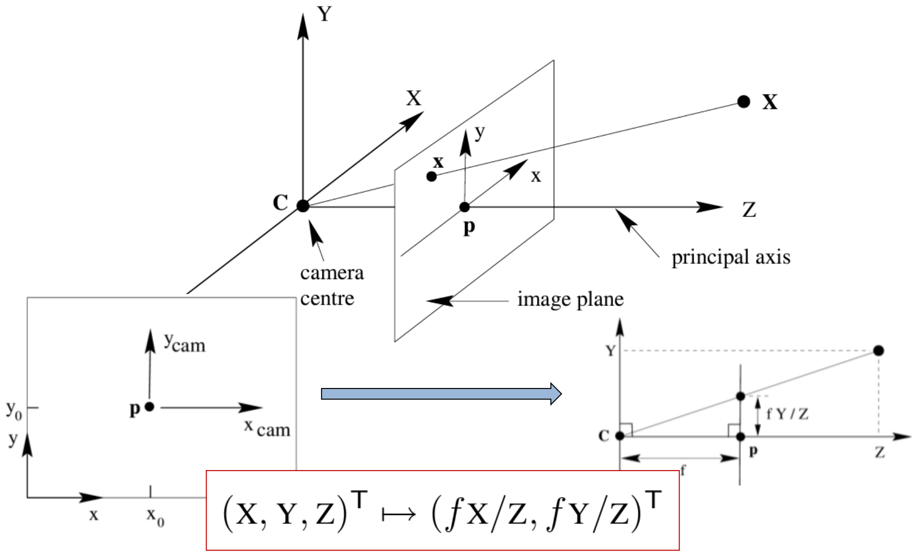
$$x=\frac{fX}{Z},\ y=\frac{fY}{Z}$$

$$(X,Y,Z)\mapsto(x,y)$$

$$\mathbb{R}^3\mapsto\mathbb{R}^2$$

Image and Slide from: Corke, Ch. 11

Image Formation – Single View Geometry [I]



$$(X, Y, Z)^{\mathsf{T}} \mapsto (fX/Z, fY/Z)^{\mathsf{T}}$$

Hartly & Zisserman, Ch. 6

Image Formation – Single View Geometry [II]

➔ Camera Projection Matrix

$$\underbrace{\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}}_{\text{world}} \mapsto \underbrace{\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix}}_{\text{camera}} = \underbrace{\begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix}}_{\text{Intrinsics}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

- x = Image point
- **X** = World point
- K = Camera Calibration Matrix

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$

$$\mathbf{x} = K[I \mid \mathbf{0}]\mathbf{X}_{cam}.$$

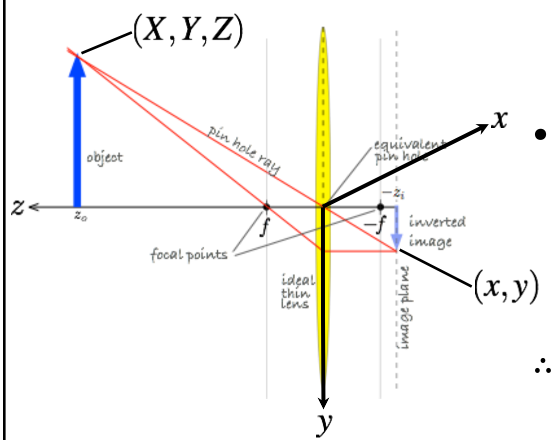
➔Perspective Camera as:

$$P = K[R \mid \mathbf{t}]$$

where: P is 3×4 and of <u>**rank 3**</u>

Image Formation: (Thin-Lens) Projection model

- $x = \frac{fX}{Z}, y = \frac{fY}{Z}$

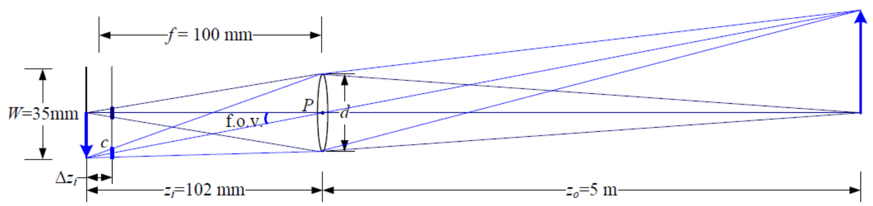


- $\frac{1}{z_0} + \frac{1}{z_1} = \frac{1}{f}$

$\therefore$ as $z_0 \rightarrow \infty$, $z_i \rightarrow f$

Image and Slide from: Corke, Ch. 11

Image Formation: Simple Lens Optics ≅ Thin-Lens



$$\frac{1}{z_0} + \frac{1}{z_1} = \frac{1}{f}$$

Sec. 2.2 from Szeliski, *Computer Vision: Algorithms and Applications*

Camera

Calibration!

Calibration matrix

- Is this form of K good enough?
- non-square pixels (digital video)
- skew
- radial distortion

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{K}\ \mathbf{X}_c$$

$$\begin{bmatrix} fa & s & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{K}$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Calibration

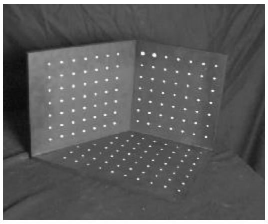
See: *Camera Calibration Toolbox for Matlab*
(http://www.vision.caltech.edu/bouguetj/calib_doc/)

- Intrinsic: Internal Parameters
  - **Focal length:** The focal length in pixels.
  - **Principal point:** The principal point
  - **Skew coefficient:**
    The skew coefficient defining the angle between the x and y pixel axes.
  - **Distortions:** The image distortion coefficients (radial and tangential distortions) (typically two quadratic functions)

- Extrinsics: Where the Camera (image plane) is placed:
  - **Rotations:** A set of 3x3 rotation matrices for each image
  - **Translations:** A set of 3x1 translation vectors for each image

Camera calibration

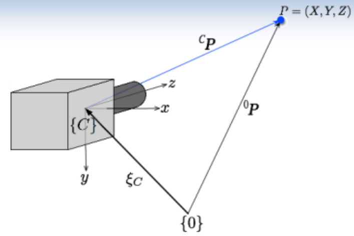
- Determine camera parameters from known 3D points or calibration object(s)
- internal or intrinsic parameters such as focal length, optical center, aspect ratio:
  what kind of camera?
- external or extrinsic (pose) parameters:
  where is the camera?
- How can we do this?



From Szeliski, *Computer Vision: Algorithms and Applications*

Complete camera model

$$\begin{pmatrix}\tilde{x}\\ \tilde{y}\\ \tilde{z}\end{pmatrix} = \begin{pmatrix}f & 0 & 0 & 0\\ 0 & f & 0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix}\begin{pmatrix}X\\ Y\\ Z\\ 1\end{pmatrix}$$



extrinsic parameters

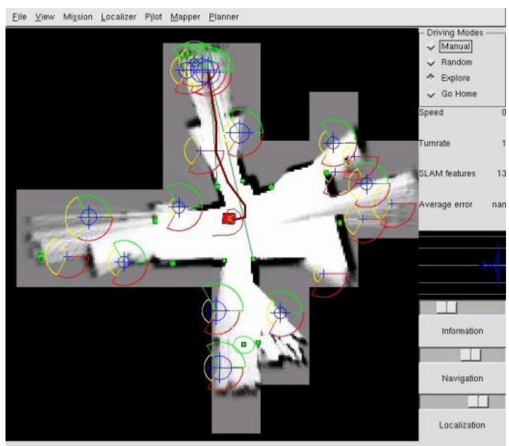
$$\begin{pmatrix}\tilde{u}\\ \tilde{v}\\ \tilde{w}\end{pmatrix} = \underbrace{\underbrace{\begin{pmatrix}\frac{1}{\rho_u} & 0 & u_0\\ 0 & \frac{1}{\rho_v} & v_0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}f & 0 & 0 & 0\\ 0 & f & 0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix}}_{\mathbf{K}}\begin{pmatrix}\mathbf{R} & t\\ \mathbf{0}_{1\times 3} & 1\end{pmatrix}^{-1}}_{\mathbf{C}}\begin{pmatrix}X\\ Y\\ Z\\ 1\end{pmatrix}$$

intrinsic parameters

camera matrix

Image and Slide from: Corke, Ch. 11

CRS: Mapping: Indoor robots



ACFR, IROS 2002