Applied Robotics
&
Course Review

METR 4202: Advanced Control & **Robotics**

Dr Surya Singh

Lecture # 11

October 25, 2013

metr4202@itee.uq.edu.au

http://itee.uq.edu.au/~metr4202/

RoboticsCourseWare.org
RoboticsCourseWare Contributor

© 2013 School of Information Technology and Electrical Engineering at the University of Queensland

(cc) BY-NC-SA

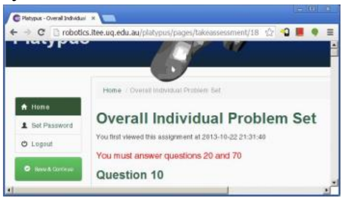
Schedule

Week	Date	Lecture (F: 9-10:30, 42-212)
1	26-Jul	Introduction
2	2-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	9-Aug	Robot Kinematics
4	16-Aug	Robot Dynamics & Control
5	23-Aug	Robot Trajectories & Motion
6	30-Aug	Sensors & Measurement
7	6-Sep	Perception / Computer Vision
8	13-Sep	Localization and Navigation
9	20-Sep	State-Space Modelling
10	27-Sep	State-Space Control
	4-Oct	*Study break*
11	11-Oct	Motion Planning
12	18-Oct	Motion Planning & Control
13	**25-Oct**	**Applications in Industry & Course Review**

Announcements: We're Working On It!

- **Lab 3:**
  - One team done"Free Extension" to October 31.

- **Take Home "Quiz":**
  - Posted! (via Platypus)
  - If you Lab 3 Score is "~$20": You may "opt-out"



- Cool Robotics Share Site
  ➔ http://metr4202.tumblr.com/

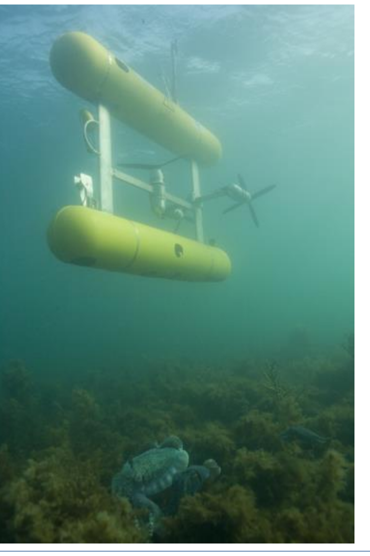
Cool Robotics Video Share

And now Presenting …
Peter Corke!

And now Presenting …
Latzi Marian!

And now Presenting …
Stefan Williams!

Robot Submarines (In South Australia !)



And now Presenting …
Hanna Kurniawati

COMP3702/7702 Artificial Intelligence

http://robotics.itee.uq.edu.au/~ai

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

The robotics design lab

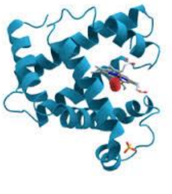
Autonomous (Robotics) Systems

Autonomous (Robotics) Systems

11

COMP3702/7702 is about

- Fundamental computational representations & techniques to enable computers:
  - Make good decisions when information about the problem is accurate & abundance.
  - Make good decisions when information about the problem is inaccurate & limited.
  - Learn & improve decision making capability over time.



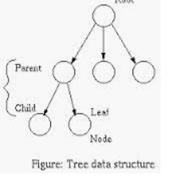
Some of the things you'll learn

- Motion planning for a high DOFs system: A (Simplified) oil containment system
  - Assignment 1

- A (simplified) stealth tracking
  - Assignment 2

Assumed background

- Understand & can implement abstract data structures, esp. trees & graphs, using a high-level programming language.



- Know basic computational complexity, e.g. big-O.

COMP3702/7702 Artificial Intelligence

http://robotics.itee.uq.edu.au/~ai

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

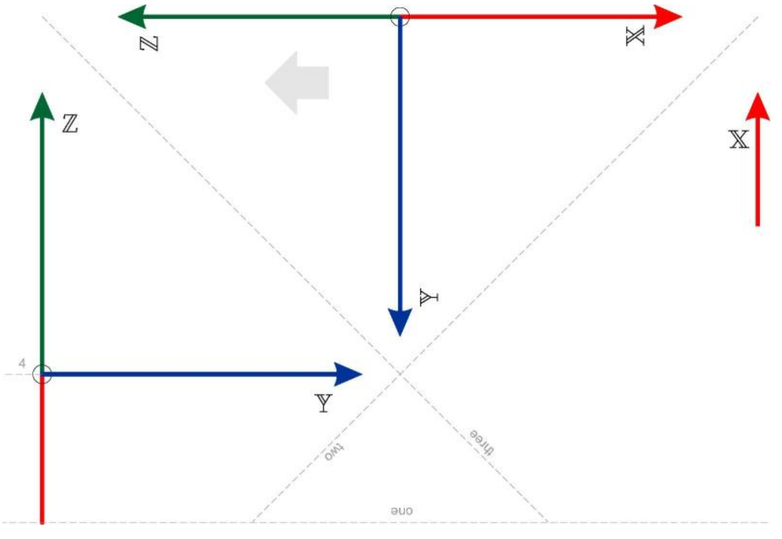
The robotics design lab

And now Presenting …
Paul Pounds!

And now Presenting …
Surya Singh (finally)!

Course Review (finally)!

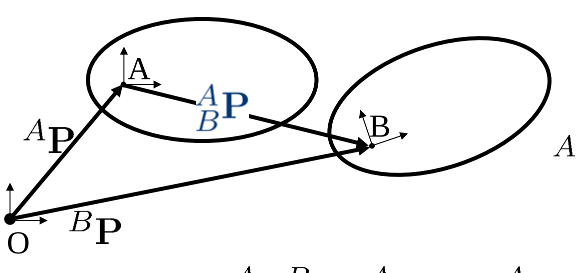
It All Starts With: **Frames in Space!!**



Position and Orientation [1]

- A **position** vectors specifies the location of a **point** in 3D (Cartesian) space

$$\mathbf{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$



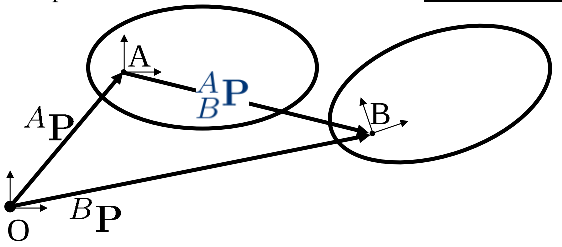
$$^{A}\mathbf{P} + {}^{A}\mathbf{P}^{B} - {}^{B}\mathbf{P} = 0$$

$$^{A}\mathbf{P}^{B} = {}^{A}\mathbf{P}_{B} = {}^{A}_{B}\mathbf{P} = \begin{bmatrix} {}^{B}p_x \\ {}^{B}p_y \\ {}^{B}p_z \end{bmatrix} - \begin{bmatrix} {}^{A}p_x \\ {}^{A}p_y \\ {}^{A}p_z \end{bmatrix}$$

- BUT we **also** concerned with its orientation in 3D space.
  This is specified as a matrix based on each **frame's unit vectors**

Position and Orientation [2]

- Orientation in 3D space:
  This is specified as a matrix based on each **<u>frame's unit vectors</u>**



- Describes {B} relative to {A}
  → The orientation of frame {B} relative to coordinate frame {A}
- Written "from {A} to {B}" or "given {A} getting to {B}"

$$^{A}\mathbf{R}_{B} = {}^{A}_{B}\mathbf{R} = \left[\begin{array}{ccc} {}^{A}\hat{\imath}_{B} & {}^{A}\hat{\jmath}_{B} & {}^{A}\hat{k}_{B} \end{array}\right]$$

- **<u>Columns</u>** are **<u>{B} written in {A}</u>**

Position and Orientation [3]

- The rotations can be analysed based on the unit components …
- That is: the components of the orientation matrix are the unit vectors projected **onto** the unit directions of the reference frame

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

$$\begin{array}{c|c} {}^{A}_{B}R & \begin{array}{ccc} (b_x)\,\hat{\imath}_B & (b_y)\,\hat{\jmath}_B & (b_z)\,\hat{k}_B \end{array} \\ \hline \begin{array}{c} (a_x)\,\hat{\imath}_A \\ (a_y)\,\hat{\jmath}_A \\ (a_z)\,\hat{k}_A \end{array} & \begin{bmatrix} \hat{\imath}_B \cdot \hat{\imath}_A & \hat{\jmath}_B \cdot \hat{\imath}_A & \hat{k}_B \cdot \hat{\imath}_A \\ \hat{\imath}_B \cdot \hat{\jmath}_A & \hat{\jmath}_B \cdot \hat{\jmath}_A & \hat{k}_B \cdot \hat{\jmath}_A \\ \hat{\imath}_B \cdot \hat{k}_A & \hat{\jmath}_B \cdot \hat{k}_A & \hat{k}_B \cdot \hat{k}_A \end{bmatrix} \end{array}$$

Position and Orientation [4]

- Rotation is orthonormal

$$\begin{array}{c|ccc} {}^{A}_{B}R & (b_x)\,\hat{i}_B & (b_y)\,\hat{j}_B & (b_z)\,\hat{k}_B \\ \hline \begin{array}{c}(a_x)\,\hat{i}_A\\ (a_y)\,\hat{j}_A\\ (a_z)\,\hat{k}_A\end{array} & \multicolumn{3}{c}{\left[\begin{array}{ccc} \hat{i}_B\cdot\hat{i}_A & \hat{j}_B\cdot\hat{i}_A & \hat{k}_B\cdot\hat{i}_A \\ \hat{i}_B\cdot\hat{j}_A & \hat{j}_B\cdot\hat{j}_A & \hat{k}_B\cdot\hat{j}_A \\ \hat{i}_B\cdot\hat{k}_A & \hat{j}_B\cdot\hat{k}_A & \hat{k}_B\cdot\hat{k}_A \end{array}\right]} \end{array}$$

- The of a rotation matrix inverse = the transpose

$$\mathbf{R}\cdot\mathbf{R}^T = 1$$

→ thus, the ***rows*** are **{A} written in {B}**

$$ {}^{B}_{A}\mathbf{R} = {}^{A}_{B}\mathbf{R}^T = {}^{A}_{B}\mathbf{R}^{-1}$$

Euler Angles

- Minimal representation of orientation (α,β,γ)
- Represent a rotation about an axis of a **moving** coordinate frame
  → ${}^{A}_{B}\mathbf{R}$: Moving frame **B** w/r/t fixed A
- The location of the axis of each successive rotation depends on the previous one! …
- So, Order Matters (12 combinations, why?)
- Often Z-Y-X:
  - α: rotation about the **z** axis
  - β: rotation about the rotated **y** axis
  - γ: rotation about the twice rotated **x** axis
- Has singularities! … (e.g., β=±90°)

Homogenous Transformation

$$\begin{bmatrix} {}^{A}R_{B} & {}^{A}p \\ \gamma & \rho \end{bmatrix}$$

- γ is a projective transformation
- ρ is a scaling value:

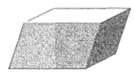
$$\hat{p} = \begin{bmatrix} \rho p_x & \rho p_y & \rho p_z & \rho \end{bmatrix}^T$$

Projective Transformations …

Group	Matrix	Distortion	Invariant properties
Projective 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, **order of contact**: intersection (1 pt contact); tangency (2 pt contact); inflections (3 pt contact with line); tangent discontinuities and cusps. cross ratio (ratio of ratio of lengths).
Affine 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (e.g. midpoints), linear combinations of vectors (e.g. centroids). The line at infinity, $\mathbf{l}_\infty$.
Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratio of lengths, angle. The circular points, $\mathbf{I}, \mathbf{J}$ (see section 2.7.3).
Euclidean 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Length, area

p.44, R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*

Projective Transformations & Other Transformations of 3D Space

Group	Matrix	Distortion	Invariant properties
Projective 15 dof	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$		Intersection and tangency of surfaces in contact. Sign of Gaussian curvature.
Affine 12 dof	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$		Parallelism of planes, volume ratios, centroids. The plane at infinity, π_∞, (see section 3.5).
Similarity 7 dof	$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$		The absolute conic, Ω_∞, (see section 3.6).
Euclidean 6 dof	$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$		Volume.

p.78, R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*

Denavit-Hartenberg: A Frame Placement Convention

1. Align an axis along principal motion
   1. <u>Rotary (**R**)</u>: align rotation axis along the <u>z axis</u>
   2. <u>Prismatic (**P**)</u>: align slider travel along <u>x axis</u>

2. Orient so as to position x axis towards next frame

3. θ $_{(\text{rot } z)}$ → d $_{(\text{trans } z)}$ → a $_{(\text{trans } x)}$ → α $_{(\text{rot } x)}$

Denavit-Hartenberg → Rotation Matrix

- Each transformation is a product of 4 "basic" transformations (instead of 6)

$$
\begin{aligned}
{}^{i-1}A_i =& Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \\
=& \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdots \\
& \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
=& \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

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

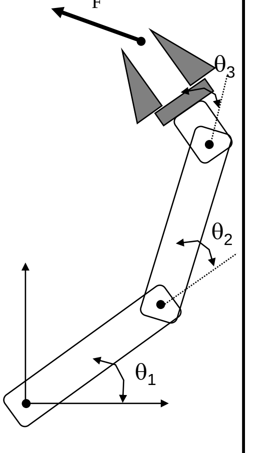
$$\delta \mathbf{X} = \mathbf{J} \delta\theta$$

- So that

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

- Or

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



Revisiting The Jacobian

- I told you (Lec 4, Slide 19):

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

- 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:

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

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

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**: $d\mathbf{p} = \mathbf{J} d\mathbf{q}$

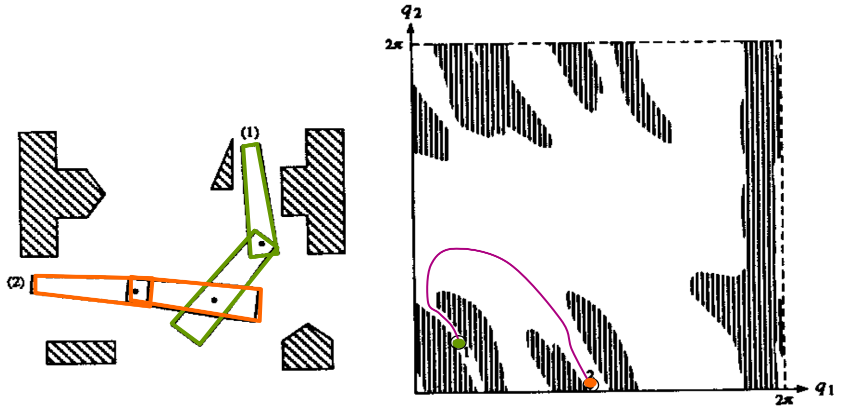
$$\mathbf{q} = \left[q_1 \quad \cdots \quad q_n\right] \rightarrow \mathbf{p} = \left[p_1 \quad \cdots \quad p_n\right]$$

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

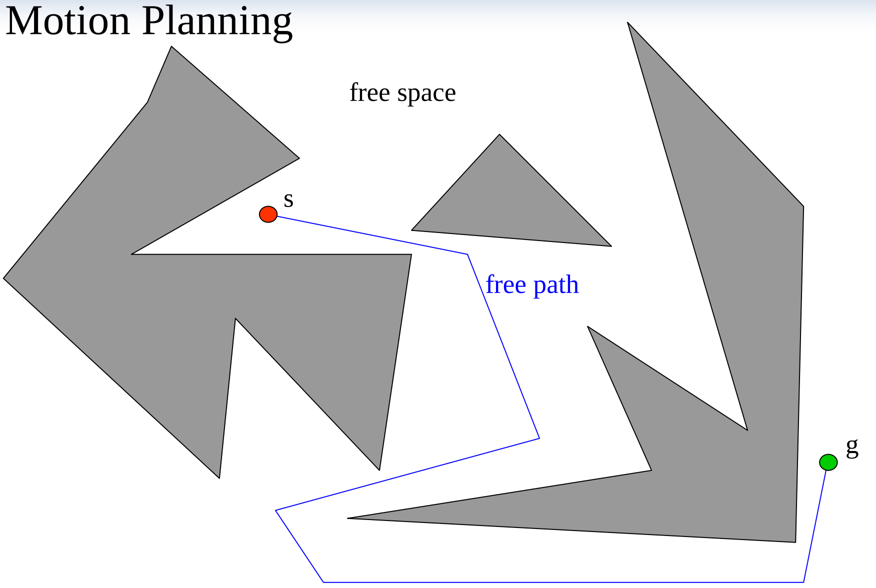
Configuration Space



- A robot configuration is a specification of the positions of all robot points relative to a fixed coordinate system
- Usually a configuration is expressed as a "vector" of position/orientation parameters

Slide from Latombe, CS326A

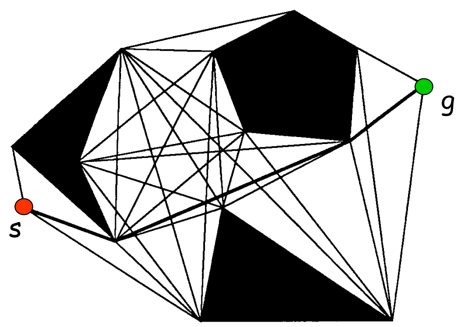
Motion Planning



Slide from Latombe, CS326A

Roadmap Methods

- **Visibility graph**
  Introduced in the Shakey project at SRI in the late 60s.
  Can produce shortest paths in 2-D configuration spaces

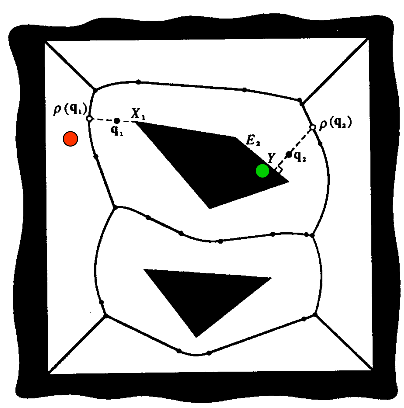


Slide from Latombe, CS326A

Roadmap Methods

- Voronoi diagram
  Introduced by Computational Geometry researchers. Generate paths that maximizes clearance.

  O(n log n) time
  O(n) space



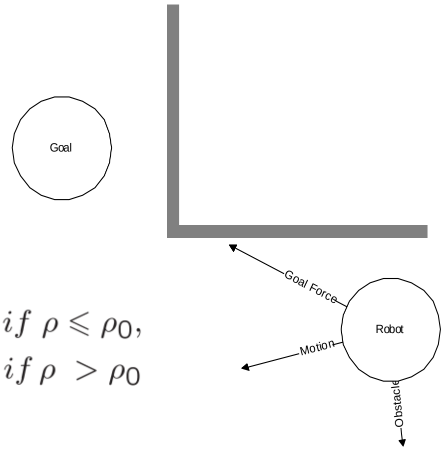
Slide from Latombe, CS326A

Potential Field Methods

- Approach initially proposed for real-time collision avoidance [Khatib, 86]

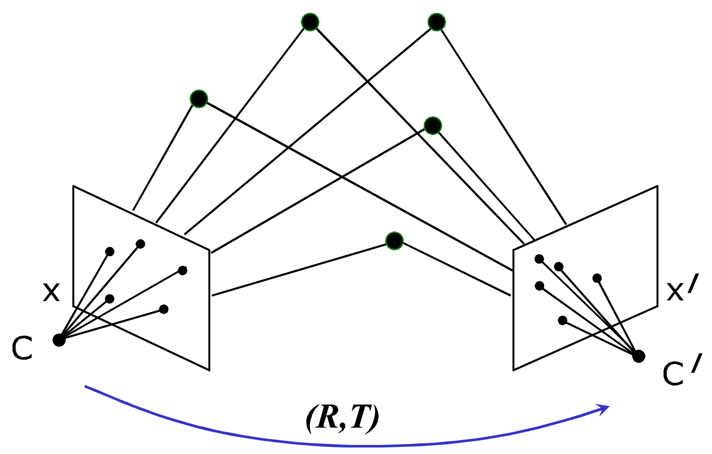
$$F_{Goal} = -k_p(x - x_{Goal})$$

$$F_{Obstacle} = \begin{cases} \eta\left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2}\frac{\partial\rho}{\partial x} & if\ \rho \leqslant \rho_0, \\ 0 & if\ \rho > \rho_0 \end{cases}$$



Slide based on Latombe, CS326A

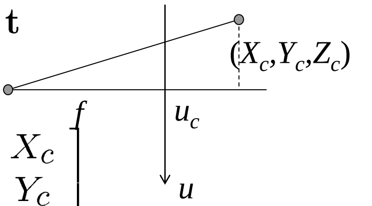
Image Formation & Steropsis



Corke, Ch. 11

Image formation equations:
3D → 2D Perspective Projection

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$



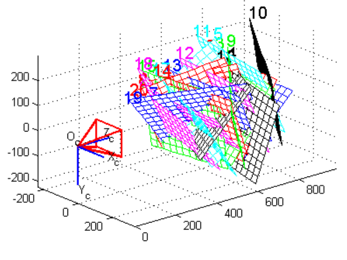
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

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

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

From Szeliski, *Computer Vision: Algorithms and Applications*

Camera calibration



- Five intrinsic parameters:
  - (f_x, f_y) : Focal length
  - (o_x, o_y) : Image center
  - α: skew
- Extrinsic Parameters:
  - Tx, Ty, Tz: Translation
  - ϕ, φ, ψ: Orientation

$$\begin{pmatrix} \tilde{X}^C \\ \tilde{Y}^C \\ \tilde{Z}^C \end{pmatrix} = R \begin{pmatrix} X^W \\ Y^W \\ Z^W \end{pmatrix} + \begin{pmatrix} T_X \\ T_Y \\ T_Z \end{pmatrix}$$

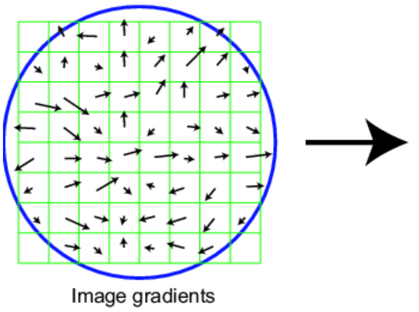
$$R = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{pmatrix}$$

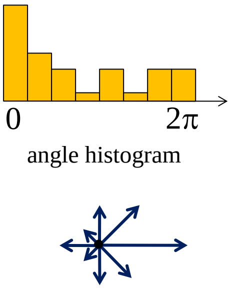
$$\begin{pmatrix} x_{im} \\ y_{im} \end{pmatrix} = \left[\begin{pmatrix} \alpha f_x \dfrac{X^C}{Z^C} + o_x + 0.5 \\ \alpha f_y \dfrac{Y^C}{Z^C} + o_y + 0.5 \end{pmatrix} \right]$$

Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations



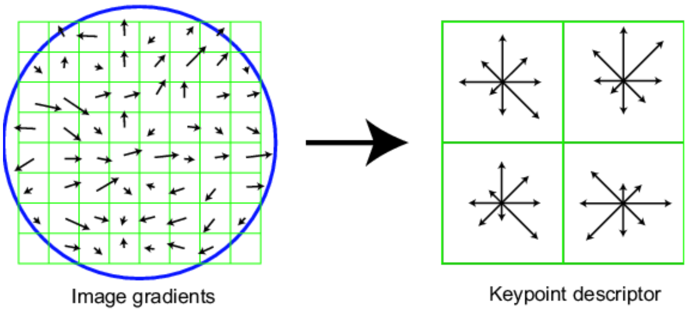


Adapted from slide by David Lowe

SIFT descriptor

Full version

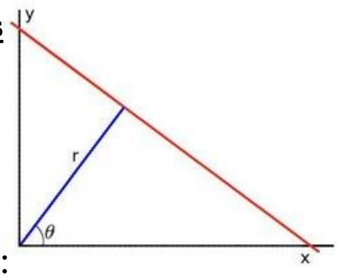
- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor



Adapted from slide by David Lowe

Line Detection – Hough Lines [1]

- A line in an image can be expressed as two variables:
  - **Cartesian coordinate system:** m,b
  - **Polar coordinate system:** r, θ
    - ➔ **avoids problems with vert. lines**



y=**m**x+**b** ➔

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$

- For each point (x_1, y_1) we can write:

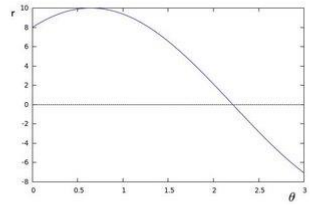
$$r = x_1 \cos\theta + y_1 \sin\theta$$

- Each pair (r,θ) represents a line that passes through (x_1, y_1)

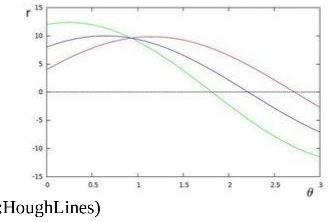
See also OpenCV documentation (cv::HoughLines)

Line Detection – Hough Lines [2]

- Thus a given point gives a sinusoid



- Repeating for all points on the image



See also OpenCV documentation (cv::HoughLines)

State-Space!

государственный контроль пространства

- *It's Russian for Control*
- **Dynamic systems** are described as differential equations (as compared to transfer functions)
- **Stability** is approached via the theory of Liapunov instead of frequency-domain methods (Bode and Nyquist)
- **Optimisation of System Performance** via calculus of variations (Pontryagin) (as compared to Wiener-Hopf methods)

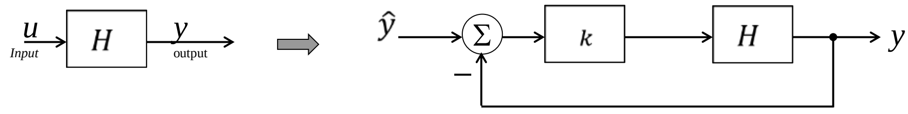
State-Space Control

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u}$$

Benefits:

- Characterises the process by systems of coupled, first-order differential equations
- More general mathematical model
  - MIMO, time-varying, nonlinear
- Mathematically esoteric (who needs practical solutions)
- Yet, well suited for digital computer implementation
  - That is: based on vectors/matrices (think LAPACK ➔ MATLAB)

Difference Equations & Feedback



- Start with the Open-Loop:

$$y = Hu$$

- Close the loop:

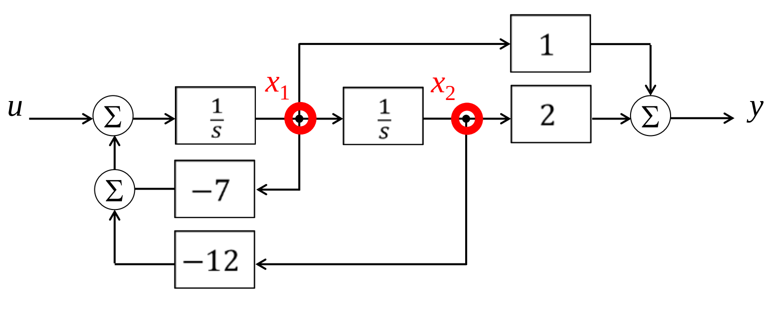
$$u = ke = k(\hat{y} - y) \rightarrow y = H[k(\hat{y} - y)]$$

$$\rightarrow y = \frac{Hk}{1+Hk}\hat{y}$$

- All easy! (yesa!)

Finding states

- We can identify nodes of a dynamic system which contain the integrated time-history values of the system response
  - We call them "states"



Linear system equations

- We can represent the dynamic relationship between the states with a linear system:

$$\begin{aligned} \dot{x}_1 &= -7x_1 - 12x_2 + u \\ \dot{x}_2 &= x_1 + 0x_2 + 0u \\ y &= x_1 + 2x_2 + 0u \end{aligned}$$

State-space representation

- We can write linear systems in matrix form:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \begin{bmatrix} -7 & 12 \\ 1 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u \\ \boldsymbol{y} &= \begin{bmatrix} 1 & 2 \end{bmatrix} \boldsymbol{x} + 0u \end{aligned}$$

Or, more generally:

$$\left.\begin{aligned} \dot{\boldsymbol{x}} &= \mathbf{A}\boldsymbol{x} + \mathbf{B}u \\ y &= \mathbf{C}\boldsymbol{x} + Du \end{aligned}\right\} \text{“State-space equations”}$$

The End… Thank you!