Robot Dynamics II: Trajectories & Motion

"Are We There Yet?"

METR 4202: Advanced Control & **Robotics**

Dr Surya Singh

Lecture # 5

August 23, 2013

metr4202@itee.uq.edu.au

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

RoboticsCourseWare.org

RoboticsCourseWare Contributor

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
	27-Sep	State-Space Control
10	4-Oct	*Study break*
11	11-Oct	Motion Planning
12	18-Oct	Vision-based control (+ Prof. P. Corke or Prof. M. Srinivasan)
13	25-Oct	Applications in Industry (+ Prof. S. LaValle) & Course Review

Announcements:

- Adam Keyes is Needing Volunteers

- UQ Summer Research Program

Outline

- Newton-Euler Formulation
- Lagrange Formulation

- Trajectory Generation & Control

- Principles of Robot Motion

First Let's Revisit The Jacobian

- Recall:

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

- 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 $\mathbf{v}_i \quad \frac{dz}{dt}$
- If the joint is **revolute** (**ϵ=0**), then $\frac{d}{dt}$ (in the $\hat{k}$ direction)

$$v \quad \sum_{i\ 1}^{N} \ {}_i v_i \quad \bar{\ }_i \quad {}_i \quad \mathbf{p}^i_{i\ 1} \qquad \sum_{i\ 1}^{N} \bar{\ }_i \ \dot{\boldsymbol{\theta}}_i \qquad \sum_{i\ 1}^{N} \bar{\ }_i \mathbf{z}_i \quad \dot{\ }_i$$

$$v \quad J_v \dot{\mathbf{q}} \qquad \boldsymbol{\omega} \quad J \ \dot{\mathbf{q}}$$

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

$$J \quad \begin{matrix} J_v \\ J \end{matrix}$$

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: ${}^F\mathbf{z}_i$ ${}^F_i R\, {}^i\mathbf{z}_i$ & ${}^i\mathbf{z}_i$ $0 \quad 0 \quad 1$

And The Inverse Jacobian

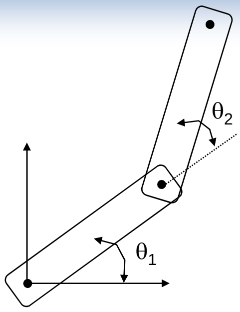
- In many instances, we are also interested in computing the set of joint velocities that will yield a particular velocity at the end effector

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

- 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

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} \\ L_1 c_1 + L_2 c_{12} & L_2 c_{12} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

- Taking the inverse of the Jacobian yields

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \frac{1}{L_1 L_2 s_2} \begin{bmatrix} L_2 c_{12} & L_2 s_{12} \\ -L_1 c_1 - L_2 c_{12} & -L_1 s_1 - L_2 s_{12} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

$-l_1 * l_2 * cos(\theta_1 + \theta_2) * sin(\theta_1) + l_1 * l_2 * sin(\theta_1 + \theta_2) * cos(\theta_1) \rightarrow l_1 * l_2 * sin(\theta_2)$

- As $\theta_2 \rightarrow 0$ (or π): it becomes singular ($s_2 \rightarrow 0$)

The Jacobian Also Relates Static **Forces & Torques**

- 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

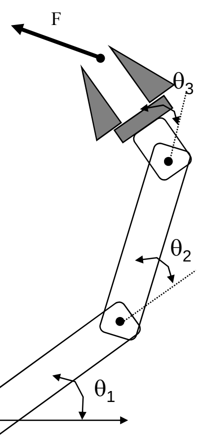
$$\delta \mathbf{X} = \mathbf{J} \delta\theta$$

- So that

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

- Or

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



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 – Newtown-Euler Mechanics

- For Manipulators, the general form is

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

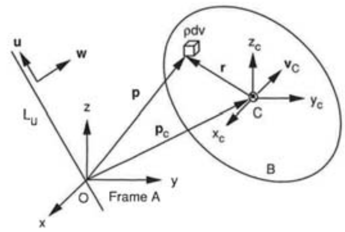
where

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

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

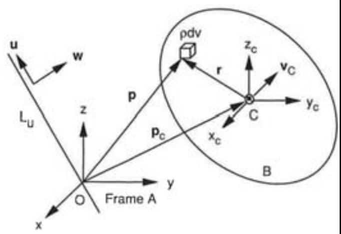
Where: $\mathbf{r_g}$: *radius of gyration* of B w/r/t to L

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

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!

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

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

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

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

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

! M is symmetric, positive definite 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]

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

$$\boldsymbol{\tau} \quad {}^{N}_{i\ 1} \quad {}_{i} \quad \frac{d}{dt}\ \frac{K}{\dot{\boldsymbol{\theta}}} \quad \frac{K}{\boldsymbol{\theta}} \quad \frac{P}{\boldsymbol{\theta}}$$

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

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

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

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

$$\boldsymbol{\tau} \quad M \quad \ddot{\boldsymbol{\theta}} \quad \mathbf{v}\ \boldsymbol{\theta},\dot{\boldsymbol{\theta}} \quad \mathbf{g}\ \boldsymbol{\theta}$$

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

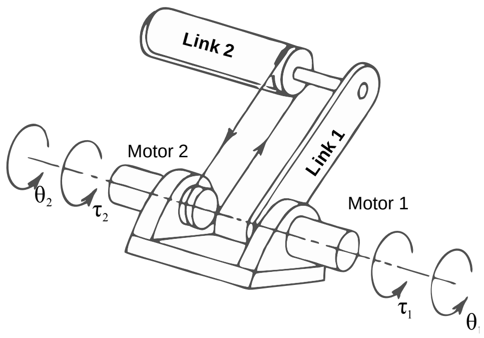
$$\mathbf{q} \quad q_1 \ \cdots \ q_n \qquad \mathbf{p} \quad p_1 \ \cdots \ p_n$$

➔Thus: the kinetic energy and gravity terms become

$$KE \quad \tfrac{1}{2}\dot{\mathbf{p}}^T\mathbf{H}^*\dot{\mathbf{p}} \qquad \mathbf{G}^* \quad \mathbf{J}^{\ 1\ T}\mathbf{G}$$

where: $\mathbf{H}^* \quad \mathbf{J}^{\ 1\ T}\mathbf{H}\mathbf{J}^{\ 1}$

Motivating Example: Remotely Driven 2DOF Manipulator



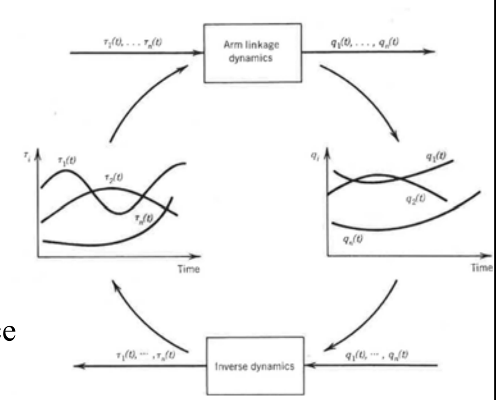
Graphic based on Asada & Slotine p. 112

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

    $\mathbf{q} \quad q_1 \quad \ldots \quad q_n \qquad {}_1\, t \quad {}_2\, t \quad {}_3\, t$
  - Outputs are joint torques to be applied at each instance

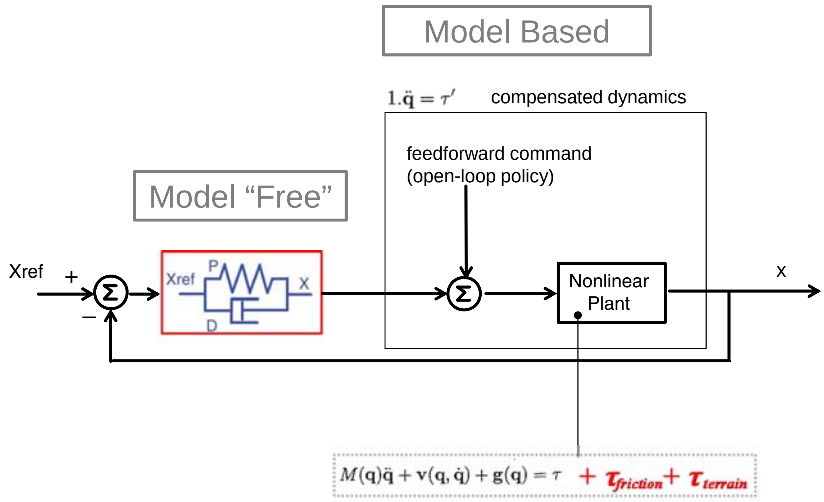
    $\boldsymbol{\tau} \qquad {}_1 \quad \ldots \quad {}_n$



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

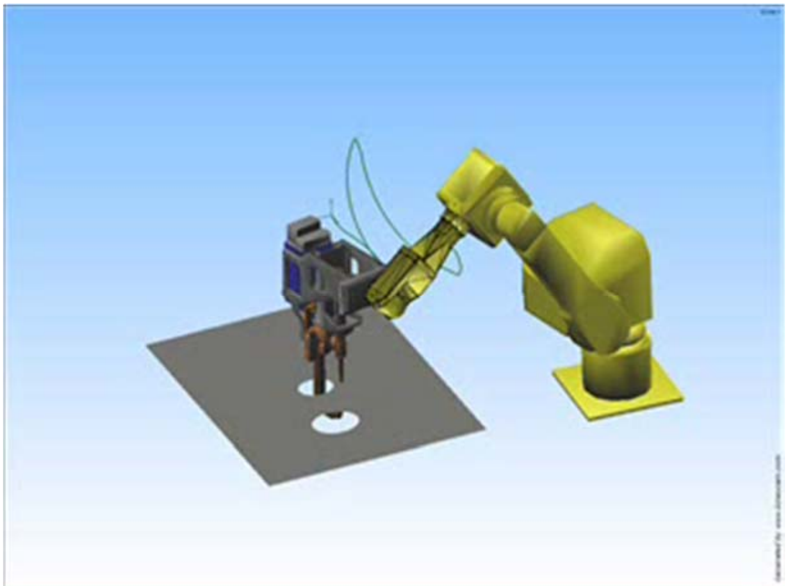
Graphic from Asada & Slotine p. 119

Operation Space (Computed Torque)



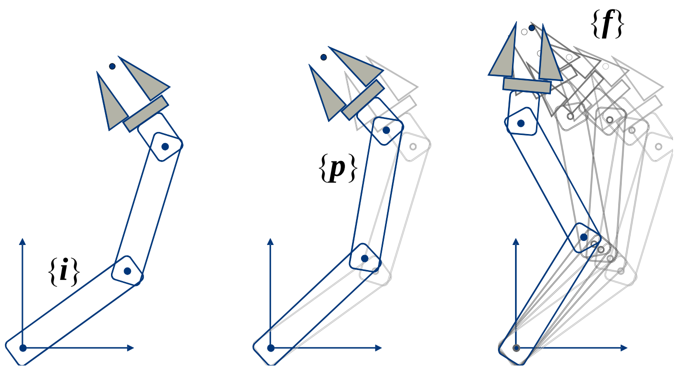
Compensated Manipulation

Trajectory Generation & Planning



Trajectory Generation

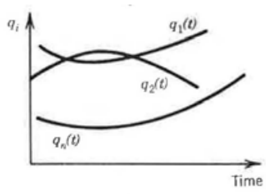
- The goal is to get from an initial position $\{\boldsymbol{i}\}$ to a final position $\{\boldsymbol{f}\}$ via a path points $\{\boldsymbol{p}\}$



Joint Space

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

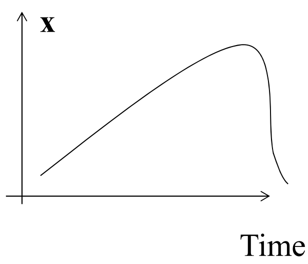
- + Since we control the joints, this is more direct
- -- If we want to follow a particular trajectory, not easy
  - at best lots of intermediate points
  - No guarantee that you can solve the Inverse Kinematics for all path points



Cartesian Workspace

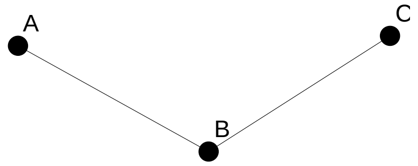
Consider the **Cartesian positions** as a function of time

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



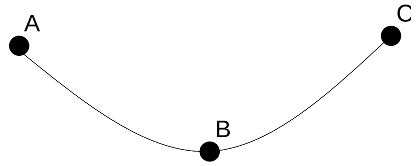
Polynomial Trajectories

- Straight line Trajectories



- Simpler

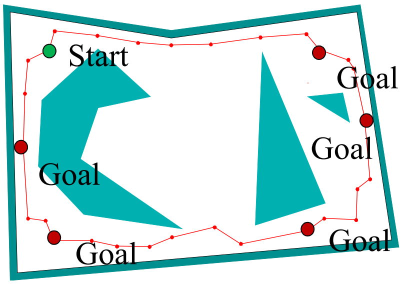
- Polynomial Trajectories



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

- Parabolic blends are smoother
- Use “pseudo via points”

Multiple Points & Sequencing



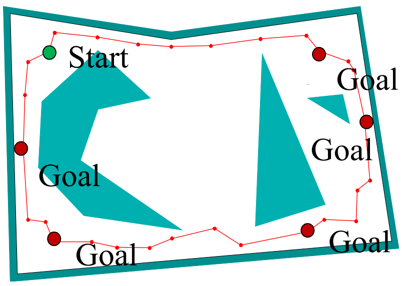
- Sequencing
  - Determining the "best" order to go in

➔ Travelling Salesman Problem

A salesman has to visit each city on a given list exactly once. In doing this, he **starts** from his home city and in the **end he has to return to his home** city. It is plausible for him to select the order in which he visits the cities so that the **total of the distances travelled** in his tour is as small as possible.

Artwork based on LaValle, Ch. 6

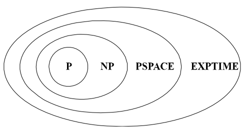
Travelling Salesman Problem



- Given a $n \times n$ distance matrix **C**=*(c$_{ij}$)*
- Minimize:

$$c(\pi) = \sum_{i=1}^{n} c_{i\pi(i)}$$

- Note that this problem is NP-Hard

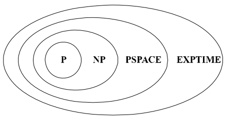


➔ BUT, Special Cases are Well-Solvable!

Artwork based on LaValle, Ch. 6

Travelling Salesman Problem [2]

- This problem is NP-Hard



➔ BUT,
Special Cases are Well-Solvable!

For the Euclidean case
(where the points are on the 2D Euclidean plane) :

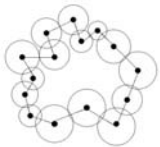
- The shortest TSP tour does not intersect itself, and thus geometry makes the problem somewhat easier.
- If all cities lie on the boundary of a convex polygon, the optimal tour is a cyclic walk along the boundary of the polygon (in clockwise or counterclockwise direction).

***The k*-line TSP**

- The a special case where the cities lie on *k* parallel (or almost parallel) lines in the Euclidean plane.
- EG: Fabrication of printed circuit boards
- Solvable in **O**(n^3) time by Dynamic Programming (Rote's algorithm)

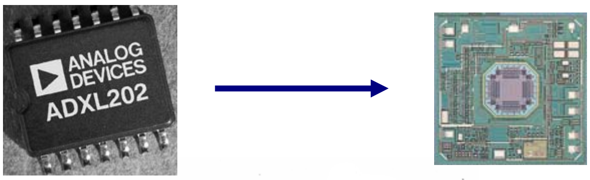
The necklace TSP

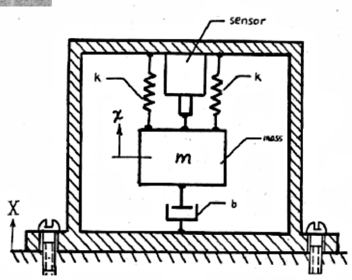
- The special Euclidean TSP case where there exist *n* circles around the *n* cities such that every cycle intersects exactly two adjacent circles



Inertial: Translation → Accelerometer

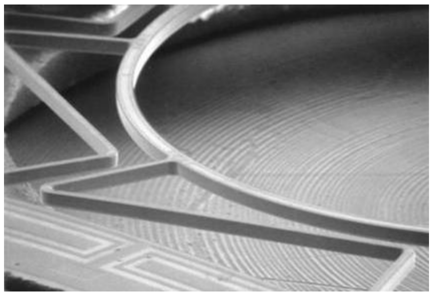
- General accelerometer:



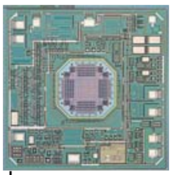


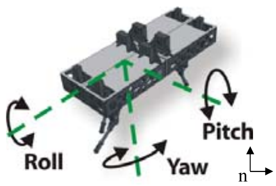
Inertial: Rotation → Gyroscopes

- Structural arrangement of silicon which records centrifugal acceleration and thus **angular speed**
- Use **strain-gauge bridges and/or piezo** structure to record deformations



Accelerometer → Acceleration



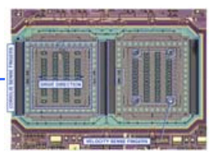


$$\vec{s} = {}^{s}_{o}\mathrm{R}\left({}^{o}_{n}\mathrm{R}\left(\mathrm{a} - \mathrm{g}\right) + \alpha \times \mathrm{r} + \omega \times \left(\omega \times \mathrm{r}\right)\right) \pm \sigma$$

signal (Accelerometer) — rotation — acceleration gravity — tangential — centripetal — Noise

$\int dt$ $\frac{\partial}{\partial t}$

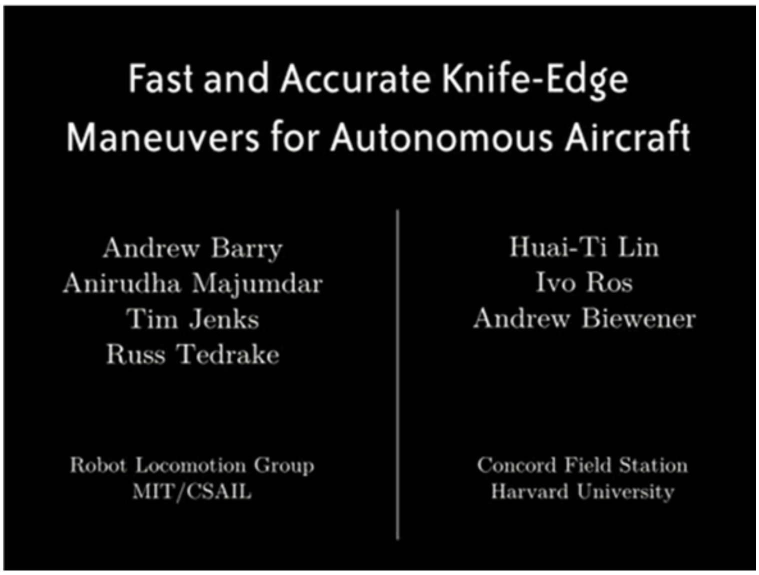
signal (Gyro)



Sources: ADI

- Noise adds uncertainty
- Gravitational & inertial forces are inseparable

Cool Robotics Share



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